Tampereen yliopisto

Tuomas Jokela

RAKENNUKSEN TIETOMALLIA HYÖDYNTÄVÄ TYÖKALU OSANA TOIMINNANOHJAUSJÄRJESTELMÄÄ

Tieto- ja sähkötekniikan tiedekunta
Diplomityö
Kesäkuu 2019

TIIVISTELMÄ

Tuomas Jokela: Rakennuksen tietomallia hyödyntävä työkalu osana toiminnanohjausjärjestelmää
Diplomityö
Tampereen yliopisto
Tietotekniikka
Kesäkuu 2019

Rakennuksen rakentaminen on valtava projekti, johon osallistuu monia tahoja eri rooleissa omine tarpeineen. Rakennusprojektiin liittyy paljon monimuotoista tietoa. Tällaista tietoa on esimerkiksi rakennuksen fyysiset ominaisuudet, aikataulu, riskienhallinta ja putkien ilmavirtaus. Jos eri tahot käyttävät ja tuottavat tällaista monimuotoista tietoa ilman mitään keskitettyä paikkaa, voi lopputuloksena olla tietokatkoksia ja ristiriitaista tietoa. Tähän ongelmaan on kehitetty ratkaisu: digitaalinen rakennuksen tietomalli eli BIM (*engl. Building Information Modeling*).

BIMin tarkoituksena on tarjota keskitetty tietovarasto kaikelle rakennuksen rakentamiseen liittyvälle tiedolle ja toimia alustana tiedonvälitykselle. Käytännössä eri tahot päivittävät ja tarkastelevat samaa BIM-mallia, joka on määritelty avoimessa ihmisen luettavassa tiedostoformaatissa. Tiedostoformaatti ei ole sovellusriippuvainen, joten sen voi avata millä tahansa sitä tukevalla ohjelmalla.

BIM-mallien käyttäminen osana rakennusprojektia on yleistynyt ja BIM-työkalujen ominaisuudet parantuneet. Kuitenkin EVRYn asiakaskyselyn mukaan Jydacom tarjouslaskenta -sovellusta käytetään yhä BIM-työkalujen ohella sen tarjoamien ominaisuuksien vuoksi. BIM-malli kuitenkin sisältää paljon tietoa, jota voisi hyödyntää tarjouslaskennasta. Tämä luo tarpeen työkalulle, jolla pystyy siirtämään tietoa BIM-mallista tarjouslaskentasovellukseen mahdollisimman automaattisesti, kätevästi ja valvotusti.

Tätä varten tässä työssä on suunniteltu ja kehitetty työkalu, joka pystyy kommunikoimaan tarjouslaskentasovelluksen kanssa ja välittämään sille haluttuja tietoja BIM-mallista. BIM-työkalu on pyritty suunnittelemaan ulkonäöltään ja suorituskyvyltään mahdollisimman samankaltaiseksi olemassa olevien BIM-sovellusten kanssa. Toiminnallisina vaatimuksina oli yleinen BIM-työkalujen toiminnallisuus, tietojen etsintä BIM-mallista sekä yhteistyövaatimukset tarjouslaskentasovelluksen kanssa. Alkuperäisenä teknologisena vaatimuksena oli, että tuotetun BIM-työkalun tulee toimia verkkoselaimessa keskustellen Windowsin työpöydällä toimivan tarjouslaskentasovelluksen kanssa.

Lopputuloksena kuitenkin päädyttiin suunnittelemaan ja toteuttamaan kaksi erillistä sovellusta: Windowsin työpöydällä toimiva BIM-työkalu, joka pystyy välittämään tietoa tarjouslaskentasovelluksen kanssa, sekä verkkoselaimessa toimiva BIM-työkalu todisteeksi konseptin toimivuudesta tulevaisuutta varten. Molemmille sovelluksille toteutettiin yhteinen logiikkakirjasto työmäärän minimoimiseksi. Jako tehtiin käytettävyyden ja tuotteistuksen parantamiseksi, vaikka verkkoselaimessa toimiva työkalun olisikin ollut teknologisesti mahdollista keskustella Windowsin työpöydällä toimivan tarjouslaskentasovelluksen kanssa.

BIM-mallien tuen mahdollistamiseksi molemmissa toteutetuissa työkaluissa käytettiin kolmannen osapuolen kirjastoa BIM-mallien tiedostojen jäsentelyyn ja niistä saadun kolmiulotteisen mallin näyttämiseen. Lopulliset sovellukset onnistuttiin tämän kirjaston avulla toteuttamaan kaikkien toiminnallisten ja ei-toiminnallisten vaatimusten mukaisesti. Kirjaston suorituskyky vastasi muiden testattujen BIM-sovellusten suorituskykyä lukuun ottamatta rakennuksen kolmiulotteisen mallin näyttämistä käyttöliittymässä.

Avainsanat: Rakennuksen tietomalli, toiminnanohjausjärjestelmä, ohjelmistosuunnittelu.

ABSTRACT

Tuomas Jokela: Integrating a software utilizing the building information modeling into the enterprise resource planning system
Master's thesis
Tampere University
Information Technology
June 2019

Construction of a facility is a significant project that has lots of stakeholders in different roles with different interests. Construction project has a huge amount of diverse information. This information could be physical characteristics of a building, work schedule, risk management information or even information about airflow in the pipes. If different stakeholders used and produce this kind of diverse information without any central database, it could lead to disappearance of information or misinformation. To avoid these problems, building information modeling or *BIM* has been developed.

The idea behind BIM is to offer a centralized database for everything associated with the construction of a facility and to offer a platform for communication between the stakeholders. In practice, different stakeholders use and update the same model, which is described in an open, human-readable format. The file format of the BIM file is not software-dependent, so it can be opened with any software supporting BIM.

BIM has been an upcoming technology and the existing BIM-software has become feature-rich. However, according to a customer questionnaire done by EVRY, Jydacom offer calculation software is still used alongside the BIM-tools for the features it offers. BIM model has information that could be useful in the offer calculation. This creates a demand for a tool that is capable of transferring information from the BIM models to the offer calculation software with ease and overseeing the process for minimization of the human errors.

This paper presents how a tool that can communicate and transfer useful information with offer calculation software can be designed and developed. The foundation of these design choices is based on the appearance and performance of the other BIM applications. Operative requirements of the tool consist of general operations seen in the existing BIM tools, data search capabilities and integration capabilities with the offer calculation software. The original technological requirement was that the developed tool needed to work in an internet browser communicating with the offer calculation software running in the Windows desktop.

However, the outcome of this research was to design and develop two different user interfaces and one business logic that both user interfaces use. One of the user interfaces was designed to run in the Windows desktop environment and communicate with the offer calculation software. The other user interfaces were designed to run in a web browser and act as a proof that the concept works for the future demands. The division was done to improve the productization and usability of the product even when it would have been possible to develop an application with only a web browser user interface for achieving the goals of this work.

For enabling the support for the BIM files, a third-party library was used in the development of both user interfaces and the logic behind them. The third-party library was used for parsing the BIM file and presenting the three-dimensional model of the building in the user interface. The final products fulfill all operative and non-operative requirements set in the beginning of the project. The performance of the existing BIM software was matched by the third-party library in all fields except in the presentation of the three-dimensional model in user interface which was considerably slower.

Keywords: Building Information Model, Enterprise resource planning, software design

ALKUSANAT

Kiitokset Jyväskylän EVRYlle diplomityöpaikkani tarjoamisesta. Kiitokset myös kaikille läheisilleni, jotka jaksoivat oikolukea näin pitkän tieteellisen tekstin.

Jyväskylässä, 4.6.2019

Tuomas Jokela

SISÄLLYSLUETTELO

1. JOHDANTO ... 1
2. TAUSTATIEDOT ... 3
2.1 BIM käsitteenä ... 3
2.2 Open BIM ... 3
2.3 Standardi IFC-dataformaatti ... 4
2.4 Verkkopohjaisen sovelluksen perusteet ... 6
2.5 Yleiskuvaus tarjouslaskentasovelluksesta ... 7
3. OLEMASSA OLEVAT BIM-OHJELMISTOT JA LIITÄNNÄISET ... 9
3.1 Olemassa olevat BIM-ohjelmistot ... 9
3.1.1 BIM Vision ... 9
3.1.2 Tekla BIMsight ... 10
3.1.3 XbimXplorer ... 11
3.1.4 Yhteenveto ... 12
3.2 BIM-liitännäiset ... 12
3.2.1 BIM Vision API ... 12
3.2.2 xBIM ... 12
3.2.3 BIMServer ... 14
3.2.4 Vertailu ja yhteenveto ... 15
4. BIM-TARKASTELUTYÖKALUN SUUNNITTELU ... 16
4.1 Vaatimukset ... 16
4.2 Toiminnallisuus ... 17
4.3 Tarkastelusovelluksessa käytettävät teknologiat ... 18
4.3.1 TypeScript ... 18
4.3.2 Angular ... 18
4.3.3 .NET-Ohjelmistokehys ... 20
4.3.4 Windows forms -kirjasto ... 20
4.4 Web-tarkastelusovelluksen arkkitehtuuri ... 20
4.5 Työpöydällä toimivan tarkastelusovelluksen arkkitehtuuri ... 23
4.6 BIM-tarkastelutyökalun näkymät ... 24
4.7 Open source -komponentin integrointi tarkastelusovellukseen ... 24
4.8 Tarkastelusovelluksen käyttöliittymä ... 25
5. BIM-TYÖKALUN LIITTÄMINEN TARJOUSLASKENTASOVELLUKSEEN ... 27
5.1 Integraation vaatimukset ... 27
5.2 Toiminnallisuus ... 28
5.3 Web-sovelluksen integrointiarkkitehtuurivaihtoehtojen vertailu ... 29
5.3.1 Peer-to-peer ... 29
5.3.2 Paikallinen tiedonsiirto ... 29
5.3.3 Viestinvälittäjä ... 30
5.3.4 Asiakasohjelma-palvelin-tietokanta-työpöytäsovellus ... 30
5.3.5 Asiakasohjelma-palvelin ... 31
5.4 Ratkaisun arkkitehtuuri ... 32
5.5 Komponentit ... 33
5.5.1 Yhteistyömoduuli ... 33
5.5.2 Vientikäsittelijä ... 34
5.5.3 Muutokset sovelluksiin ... 34

5.6Vientikäsittelijän käyttöliittymä 34
6. TOTEUTUSPROSESSI 36
6.1Työkalut ja työtavat 36
6.2Prosessin kuvaus ja tulokset 36
6.3Jatkokehitys 38
7. OHJELMAN ARVIOINTI 39
7.1Vertailu alkuperäisiin vaatimuksiin 39
7.2Vertailu olemassa oleviin ohjelmiin 39
7.3Vaihtoehtoiset toteutustavat 40
7.4Kilpailu ja vaihtoehtoiset ohjelmat 41
7.5Ohjelman ja teknologian tulevaisuus 41
8. YHTEENVETO 42
LÄHTEET 44

LYHENTEET JA MERKINNÄT

BIM	engl. Building Information Modeling, rakennuksen tietomalli
CAD	engl. Computer Aided Design, tietokoneen käyttöä suunnittelutyössä apuvälineenä
HTTP	engl. Hyper Text Transfer Protocol, protokolla, jota selaimet ja WWW-palvelimet käyttävät tiedonsiirtoon
HTML	engl. Hypertext Markup Language, verkkosivuissa yleisesti käytetty merkkauskieli
IP-Address	engl. Internet Protocol Address, numeerinen osoite verkossa olevalle laitteelle
IFC	engl. Industry Foundation Classes, rakentamisen informaatio digitaalisessa muodossa
MVC	engl. Model-View-Controller, arkkitehtuuri, jossa on eroteltu logiikka, ohjaus ja käyttöliittymä
REST	engl. Representational State Transfer, arkkitehtuurityyli, joka määrittelee rajoitteita verkkopalveluiden luomiselle

1. JOHDANTO

Rakennuksen rakentaminen on iso ja kompleksi kokonaisuus, johon osallistuu monia eri tahoja, kuten omistajat, suunnittelijat, alihankkijat ja rakentajat. Vaikka rakennuksen kolmiulotteisen mallin suunnitteluun on monia vuosia käytetty CAD [11, s. 183] (*engl. Computer Aided Design*) -ohjelmistoja, usein projektin eri osastot ovat olleet hyvin lokeroituja ja niiden välillä välitetty tieto on voinut olla parhaimmillaan printattuja piirroksia ja määritelmiä.

Usea projekti on kärsinyt suuria kustannustappioita osakkaiden välisen huonon yhteistoiminnan ja tiedonvälityksen ongelmien vuoksi. Ongelma on ollut siinä, että tieto on ollut erittäin hajautettua, eivätkä kaikki projektin osapuolet ole päässeet siihen käsiksi tai hyödyntämään sitä. Myös tarve automatisoida ja visualisoida raskasta ja monimutkaista rakennusprosessia on kasvanut alan kasvun ja teollisuuden automatisoitumisen myötä. Näitä ongelmia varten on kehitetty ratkaisu: rakennuksen tietomalli eli BIM (*engl. Building Information Modeling*). [31]

Käyttäen BIMin periaatteita ja käytäntöjä kaikkea rakennusprojektiin liittyvää tietoa voidaan hallita ja välittää digitaalisesti. Täysin itsenäinen digitaalinen esitysmuoto talon tai laitoksen rakentamisprojektista auttaa esimerkiksi visualisoimaan, muokkaamaan ja testaamaan suunnitelmaa virtuaalisesti [31]. Monet rakennusyritykset käyttävätkin jo nykypäivänä BIMin periaatteita rakennusprojekteissaan ja luovat digitaalisia malleja ja suunnitelmia rakennuksista [5]. BIMin luomiseen ja muokkaamiseen on olemassa monia erilaisia kaupallisia työkaluja, joita yritykset jo yleisesti käyttävät, esimerkiksi Autodesk Revit [3].

Vaikka rakennuksen tietomalli on hyvin kokonaisvaltainen, sitä käyttävät sovellukset eivät yksinään pysty kaikkeen toiminnallisuuteen mitä rakennusprosessissa tarvitaan. Esimerkiksi EVRYn tekemän asiakaskyselyn mukaan moni asiakas käyttää BIM-työkalujen ohella myös EVRYn Jydacom-tarjouslaskentasovellusta tarjouslaskentaan [13]. Rakennuksen tietomalli kuitenkin sisältää hyvin paljon tietoa, jota voitaisiin hyödyntää esimerkiksi juuri kyseisessä tarjouslaskentasovelluksessa. Tähän mennessä tietoa on siirretty näiden sovellusten välillä manuaalisesti, joka tietenkin lisää työmäärää ja virhealttiutta. Näiden haittojen takia tämä prosessi olisi hyvä saada digitalisoitua ja mahdollisimman automatisoiduksi.

Haasteeksi kuitenkin muodostuu tiedon siirtäminen, sillä rakennuksen tietomalli ja tarjouslaskentatyökalu eivät sisällä suoraan yhteensopivaa informaatiota. Tämän vuoksi tietoa ei voida siirtää täysin automaattisesti. Tämä työ keskittyy tutkimaan ja ratkaisemaan tämän ongelman suunnittelemalla ja toteuttamalla lisäohjelman, joka toimii kyseisen tarjouslaskentasovelluksen kanssa saumattomassa yhteistyössä auttaen käyttäjää siirtämään rakennuksen tietomallin tietoja tarjouslaskentasovellukseen mahdollisimman automatisoidusti.

Tässä työssä suunniteltavassa toteutettavassa lisäohjelmassa käyttäjälle visualisoidaan rakennuksen tietomallista saatu kolmiulotteinen malli ja rakenneobjektihierarkia, josta käyttäjä voi kätevästi lajitella ja valita haluamansa objektit. Tarkoituksena on tuottaa sovelluksesta itsenäinen kokonaisuus, joka pystyy lähettämään ja vastaanottamaan tietoa halutussa muodossa saumattomasti Windowsin työpöydällä toimivaan tarjouslaskentasovelluksen kanssa. Suunnittelussa painotetaan käyttäjäkokemusta, visuaalista selkeyttä ja prosessin nopeutta, sillä tarkoituksena on tuottaa ohjelma, joka toimii tarjouslaskentasovelluksen käytön apuna ja nopeuttaa prosessia.

Alkuperäisenä tarkoituksena oli suunnitella lisäohjelma jo olemassa olevan web-portaalin itsenäiseksi moduuliksi. Kuitenkin suunnittelun edetessä huomattiin verkkopohjaisen sovelluksen integroinnin vaikeus nykyiseen tarjouslaskentasovellukseen web-portaalin ja tarjouslaskentasovelluksen ollessa erillisiä ja toisistaan riippumattomia tuotteita. Tämän vuoksi päädyttiin jakamaan suunniteltu rakennuksen tietomallia hyödyntävä työkalu kahteen osaan: verkkopohjaiseksi sovellukseksi, joka toimii todisteena konseptin toimivuudesta verkkopohjaisena ja työpöytäsovel-

lukseksi, joka liitetään osaksi tarjouslaskentasovellusta. Molemmat sovellukset suunnitellaan siten, että niitä pystyy käyttämään itsenäisesti, mutta kuitenkin mahdollistaen niiden integroinnin toiseen ohjelmaan tai isompaan kokonaisuuteen.

Tässä työssä pyritään vastaamaan tutkimuskysymyksiin: ”miten on mahdollista toteuttaa rakennuksen tietomallia hyödyntävä työkalu, josta pystyy välittämään halutun tiedon rakennuksen tietomallista tarjouslaskentasovellukseen”, ”miten rakennuksen tietomallia hyödyntävän työkalun tulee toimia, jotta se olisi mahdollisimman käytettävä ja visuaalisesti selkeä” ja ”mitä muutoksia täytyy tehdä tarjouslaskentasovelluksessa olevaan arkkitehtuuriin, jotta se pystyy käsittelemään rakennuksen tietomallia hyödyntävältä työkalulta tullutta tietoa muuttamalla mahdollisimman vähän olemassa olevaa teknistä toteutusta”.

Työssä tutkitaan rakennuksen tietomallia, sen käyttöä ja sen merkitystä nykypäivänä luvussa 2 kirjallisuuskatsauksen muodossa. Työssä analysoidaan ja tutkitaan nykyisiä BIM-työkaluja ja BIM-ohjelmointikirjastoja luvussa 3. Luvussa 4 suunnitellaan BIM-työkalun rakenne ja luvussa 5 suunnitellaan sen liittäminen laajempaan kokonaisuuteen eli tarjouslaskentasovellukseen. Luvussa 6 kuvataan työprosessi ja sillä saavutettu lopputulos. Tämän jälkeen luvussa 7 arvioidaan työn lopputuloksia ja verrataan niitä alkuperäisiin vaatimuksiin, jotka on asetettu edellisissä luvuissa.

2. TAUSTATIEDOT

Rakennuksen tietomalli eli BIM on virtuaalinen kopio tai suunnitelma oikeasta rakennuksesta ja sen elinkaaresta [19, s. 448]. Sen toteuttamista ja visualisointia varten on olemassa erilaisia työpöytä- ja verkkopohjaisia sovelluksia, joista osa on maksullisia, osa ilmaisia ja osa avointa lähdekoodia. Koska erilaisia ja eroavia sovelluksia ja käyttötarkoituksia on niin paljon, on kehitetty avoin standardi openBIM, jonka ideana on yhtenäistää erilaiset rakennuksen tietomallin formaatit ja esitystavat. OpenBIMiä ajava yritys BuildingSMART on standardoinut rakennuksen tietomallille yhtenäisen ja ihmisen luettavan tietomuodon IFC (*engl. Industry Foundation Classes*). [9]

2.1 BIM käsitteenä

BIM on ollut jo jonkin aikaa hyvin pinnalla oleva käsite rakennustekniikassa ja se on aikaisemmin tarkoittanut paljon eri asioita eri alojen ammattilaisille. Joillekin BIM on ollut tietokoneohjelmisto, kun taas toisille se on ollut rakennuksen suunnittelun ja dokumentoinnin prosessi. Osalle rakennusprojektiin osallistujista se on myös ollut tapa hallita uusia sääntöjä, alihankkijoita ja suhteita projektin osakkaiden kesken [1, s. 420]. Käsitteen yhtenäistämiseksi Yhdysvaltojen ”The National Building Information Model Standard Project Committee” on standardoinut BIMin olevan digitaalinen esitys laitoksen fyysisistä ja funktionaalisista ominaisuuksista, joka toimii jaettuna tiedonlähteenä laitoksesta luoden luotettavan pohjan laitosta koskeville päätöksille sen koko elinkaaren aikana, mielikuvasta suunnitteluun ja valmiiseen rakennukseen [31].

BIM on virtuaalinen esitys fyysisestä rakennuksesta, joten kaikkein ilmiselvin data mitä BIM sisältää on rakennuksen geometriset ominaisuudet kuten seinät, ikkunat sekä niihin liittyvät sijainnit ja koot. Geometrinen data on kuitenkin vain pieni osa BIMin sisältöä, sillä se sisältää geometrisen datan lisäksi paljon projektin eri osallistujia hyödyttävää dataa, kuten käytettävä rakennusmateriaali ja ilmastointiputkien ilmavirtaus. BIM ei yleensä valmistu kerralla projektin alussa, vaan se elää ja päivittyy koko projektin elinkaaren ajan mukautuen muutoksiin. BIM on yleiskäsite rakennuksen tietomallista, eikä sillä ohjelmistokehityksen näkökulmasta ole vain yhtä tiedostomuotoa. [19, s. 448]

Kaiken aineellisen informaation lisäksi BIM voi sisältää rakennusprojektia koskevaa aineetonta informaatiota, esimerkiksi rakennuksen aikataulun ja tietoa riskienhallinnasta. Ensinäkemältä BIM voi vaikuttaa rakennuksen tietojen tietosäiliöltä, mutta todellisuudessa se on oliopainotteinen suunnitelma rakennuksesta, sillä se esittää kaikki tietorakenteet olioina (esimerkiksi seinät, ovet ja ikkunat). Oliot sisältävät tietoa niiden ominaisuuksista ja suhteista toisiin olioihin. [4, s. 540]

Visualisoinnin ja informaation välityksen lisäksi BIMiä voi hyödyntää automaattisena rakennuksen turvatarkistuksena. Rakennusprosessi voi esimerkiksi sisältää puutteellisia tai vaarallisia aktiviteettisarjoja, jotka BIM voi automaattisesti tunnistaa ja eliminoida heti rakennuksen suunnitteluvaiheessa. Näin ollen virheellisesti suunnitellut rakenteet ja osat voidaan tunnistaa jo suunnitteluvaiheessa tarkastelemalla rakennuksen geometrista mallia. [45, s. 183-184]. BIM:iä voi myös käyttää esimerkiksi nopeuttamaan ja automatisoimaan aikataulutusta ensin hakemalla mallista paikkatieto, materiaalien laatu ja määrä, sitten generoimalla aktiviteetit ja niiden kestot perustuen haettuihin tietoihin ja tämän jälkeen kehittämällä järjestyksen mainituille aktiviteeteille ja näistä generoimalla lopullisen optimaalisen aikataulun [21].

2.2 Open BIM

Edellisessä aliluvussa esiteltyjen BIMin käyttötarkoituksia kuvaavien esimerkkien perusteella voidaan nähdä, että BIMillä on potentiaalia hyödyttää jokaista rakennusprojektiin osallistuvaa osapuolta, kunhan sitä on saumaton käyttää projektin eri osallistujien välillä. Ongelmaksi kuitenkin muodostuu eri ohjelmistot ja eri käyttötarkoitukset kyseiselle tietomallille, mikä vaikeuttaa yhteistoimintaa eri osakkaiden välillä ja aiheuttaa ylimääräisiä kustannuksia projektille sekä hidastaa

projektin etenemistä. NIST-tutkimuksessa arvioitiin, että huonoon yhteistoimintaan projektin omistajalla kuluu keskimäärin vuosittain noin 15,8 biljoonaa dollaria [31].

Tähän yhteistoimintaongelmaan ratkaisuna on avoin standardi openBIM, jonka ideana on, että riippumatta käytetystä ohjelmasta ja osakkaan roolista rakennuksen tietomalli on saatavilla, nähtävissä ja muokattavissa. OpenBIM on käsite, joka keskittyy siihen, että tietomalli on riippumaton tietystä ohjelmistosta tai ohjelmistoperheestä ja varmistaa, että tietomalli toimii saumattomasti eri ohjelmien välillä. [20, s. 2]

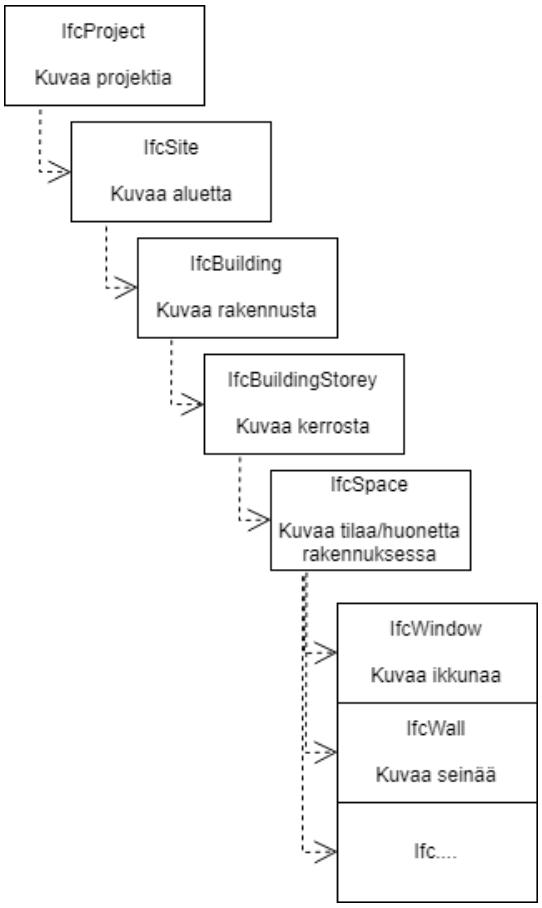
OpenBIMissä rakennuksen kaikki tiedot, mukaan lukien aineelliset, aineettomat ja niiden suhteet, ovat avoimesti luettavassa muodossa, jonka mikä tahansa ohjelma tai ihminen voi lukea [19, s.449]. Näin tietomalli ei ole lukittu yhteen ohjelmaan tai ympäristöön. OpenBIMin tavoitteena on parantaa rakennusprosessin tuottavuutta ja tehokkuutta. Sen käyttö voi hyödyttää eri projektiin osallistujia eri tavalla. OpenBIMin avulla esimerkiksi rakennusprojektin omistaja voi löytää ja päästä käsiksi kaikkeen projektiin liittyvään informaatioon turvallisesti, kun taas toimittajat voivat vaihtaa sen avulla kätevästi laitokseen liittyvää informaatiota digitaalisesti ja kaikki projektiin osallistuvat voivat tarkastella suunniteltua taloa kolmiulotteisesti haluamallaan ohjelmistolla [31]. BuildingSMART on kehittänyt yleisesti käytetyn openBIM-standardin IFC tätä tarkoitusta varten [20]. IFC 4 -version jälkeen IFC on ollut openBIM ISO 16739 -standardi [8].

2.3 Standardi IFC-dataformaatti

IFC on BuildingSMARTin kehittämä ja ylläpitämä kansainvälinen openBIMin standardi dataformaatti. Se on avoin määritelmä rakennusprojektiin osallistujien ja hallinnon välillä [7]. IFC sisältää tiedon siitä miltä rakennus näyttää, miten sitä käytetään ja miten se rakennetaan. IFC voi määritellä esimerkiksi talojen fyysisiä komponentteja, valmistettuja tuotteita, mekaanisia tai sähköisiä järjestelmiä sekä työn aikataulua [21, 22], s. 184. IFC-formaatin käyttäjät voivat itse määritellä miten jakavat ja käyttävät tietoja. IFC-standarditiedostoja voidaankin määritellä erilaisissa dataformaateissa kuten XML, JSON tai STEP. Luotuja IFC-tiedostoja voi jakaa osakkaiden kanssa esimerkiksi verkon avulla tai säilyttää tietokannoissa. [9]

Talon geometria on määritelty IFC-tiedostossa verkostoissa, jotka muodostuvat eri kokoisista ja muotoisista kolmioista. Niiden avulla IFC-tiedosto pystyy määrittelemään minkä tahansa käyrän tai suoran kolmiulotteisen pinnan. IFC:ssä on myös sisäänrakennettu apu määrittelemään talon tyypillisiä osia kuten seiniä, ikkunoita ja ovia. Tiedoston objekteja voi myös helposti laajentaa mukautetuilla ominaisuuksilla, tehden formaatista hyvin mukautuvan ja laajennettavan omiin käyttötarkoituksiin. [9]

Kaikki data on esitetty IFC:ssä hierarkkisessa muodossa, jossa kaikki objektit periytyvät *IfcRoot*-luokasta, jonka ominaisuuksiin kuuluu, että jokaisella objektilla on aina oma uniikki elinkaaren ajan säilyvä 128-bittinen tunniste, ihmisen luettava nimi ja kuvaus. IFC pystyy pitämään sisällään yksittäisiä objektin ilmentymiä, kuten esimerkiksi tietty ikkuna tietyssä talossa. Näistä yksittäisistä objekteista voi tiedostossa muodostua avaruudellinen hierarkia, joka kuvaa rakennusta kolmiulotteisesti. Esimerkki tällaisesta hierarkiasta on esitetty kuvassa 1. [9]



Kuva 1. Esimerkki IFC-objektihierarkiasta

Vaikka jokainen IFC-tiedosto on ihmisen luettavaa tekstiä, se ei tarkoita, että tietomalleja olisi helppo muokata suoraan tiedostosta esimerkiksi tekstieditorilla. Tiedostot muodostuvat helposti moniksi tuhansiksi riveiksi määrittelyjä jo yksinkertaisissakin rakennuksissa. Esimerkiksi yhden itsenäisen seinän ja sen projektin määrittelemiseen tarvitsee noin 60 riviä määrittelyä. Ohjelmasta 1 voidaan nähdä yksinkertaisen seinän geometrian määrittely IFC-formaatissa.

```
#303= IFCWALL('0DWgwt6o1FOx7466fPk$jl',#56,$,$,$,#306,#318,$,$);
#304= IFCMATERIALLAYERSETUSAGE(#210,.AXIS2.,.POSITIVE.,0.0,$);
#306= IFCLOCALPLACEMENT($,#307);
#307= IFCAXIS2PLACEMENT3D(#2,$,$);
#308= IFCCARTESIANPOINT((5000.0,0.0));
#309= IFCCARTESIANPOINT((0.0,0.0));
#310= IFCPOLYLINE((#309,#308));
#311= IFCSHAPEREPRESENTATION(#5,'Axis','Curve2D',(#310));
#313= IFCAXIS2PLACEMENT2D(#314,$);
#314= IFCCARTESIANPOINT((2500.0,135.0));
#315= IFCDIRECTION((0.0,0.0,1.0));
#316= IFCEXTRUDEDAREASOLID(#312,$,#315,2000.0);
#317= IFCSHAPEREPRESENTATION(#5,'Body','SweptSolid',(#316));
#318= IFCPRODUCTDEFINITIONSHAPE($,$,(#311,#317));
```

Ohjelma 1. Esimerkki IFC-määrittelystä

Ohjelmassa 1 aluksi määritellään seinä, siihen liittyvät materiaalit ja muut ominaisuudet, jonka jälkeen määritellään avaruudelliset ominaisuudet pisteiden ja niiden välisten viivojen avulla. Monet rakennuksen tietomallin käsittelyohjelmat tukevat jo IFC-formaattia ja tarjoavat työkaluja sen käsittelylle ja muokkaamiselle. Tällaisia ohjelmia ovat esimerkiksi ilmainen BimVision [12] ja maksullinen Autodesk Revit [3]. Osa ohjelmista myös perustuu jo täysin IFC-formaattiin, eivätkä

tarjoa edes omaa formaattiaan. Tällainen ohjelma on esimerkiksi avoimen lähdekoodin BimServer [6].

2.4 Verkkopohjaisen sovelluksen perusteet

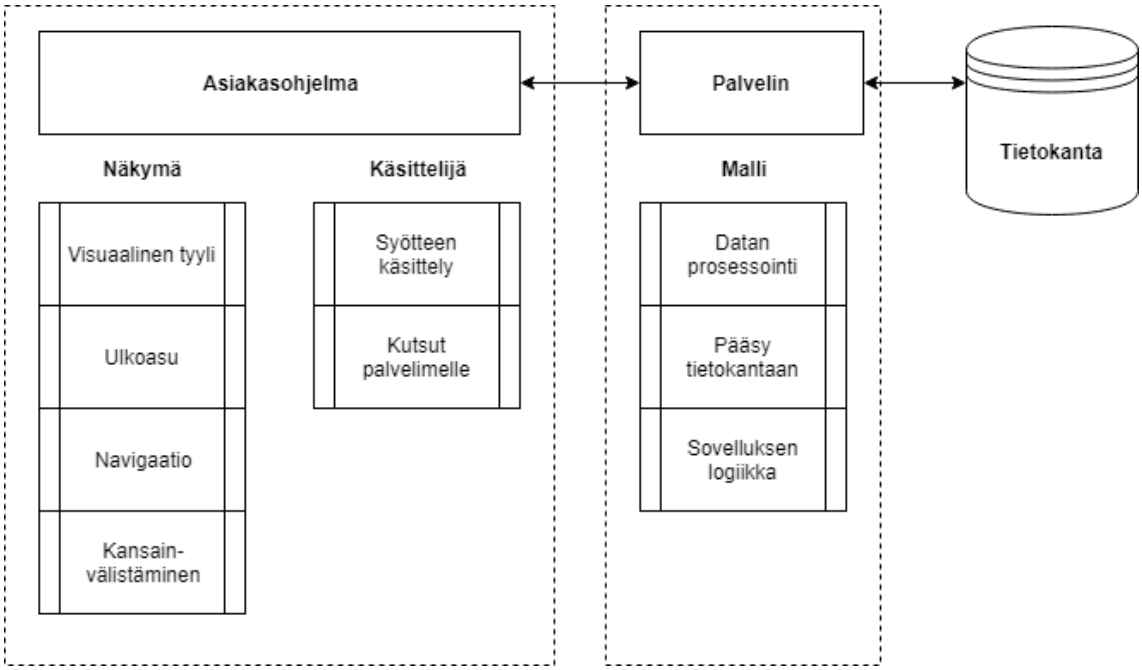
Toinen tässä työssä suunniteltavista ja toteutettavista sovelluksista tulee toimimaan verkossa BIM-mallin tarkastelua varten ja se toimii eräänlaisena tulevaisuuden referenssinä mahdollista BIM-teknologian laajempaa verkkosovelluksiin integrointia varten. Tämä sovellus tullaan liittämään omaksi itsenäiseksi moduulikseen web-portaaliin.

Jo 90-luvulta asti olemassa olleet verkkosovellukset ovat pystyneet tarjoamaan toiminnallisuutta käyttäjille ilman, että käyttäjän on tarvinnut huolehtia mistään muusta kuin ajan tasalla olevasta verkkoselaimesta. Suurin osa kompleksista toimintalogiikasta on tietoa käyttäjälle välittävällä palvelimella. Vuosien varrella verkkopohjaisten sovellusten ominaisuudet, luotettavuus, ylläpidettävyys ja skaalautuvuus ovat parantuneet huomattavasti. [23, s. 80]

Tyypillisesti modernissa verkkosovelluksessa on kolme osaa: käyttöliittymä, bisneslogiikka ja tietojen säilytys. Yleisesti nämä toimivat yhdessä malli-näkymä-käsittelijä eli MVC (*engl. Model-View-Controller*) arkkitehtuurin mukaisesti, jossa malli on vastuussa kaikesta tiedon prosessoinnista ja välityksestä, näkymä on vastuussa visuaalisesta esityksestä käyttöliittymässä ja käsittelijä on vastuussa syötteen validoinnista ja välityksestä mallille. [23, s. 80-81]

Verkkosovellus jakautuu yleisesti kahteen eri prosessiin: palvelin ja asiakasohjelma, jotka sijaitsevat yleensä fyysisesti eri laitteilla. Verkkosovelluksen näkymä ja käsittelijä sijaitsevat käyttäjän tietokoneen selaimessa olevan asiakasohjelman puolella. Verkkosovelluksen logiikka ja tietokanta sijaitsevat palvelimen puolella. Asiakasohjelma on yleensä vastuussa käyttäjälle näkyvästä käyttöliittymästä, käyttäjän antaman syötteen validoinnista ja käsittelystä sekä tietojen pyytämisestä palvelimelta. Asiakasohjelma pystyy tekemään pyyntöjä palvelimen ohjelmointirajapinnan eli APIn (*engl. Application interface*) päätepisteille esimerkiksi halutessaan hakea tietoa tietokannasta tai halutessaan tallentaa tietoa tietokantaan. Palvelin on yleensä vastuussa käyttäjien luvista, palvelun eheydestä ja rajoitteista, tietokantapyynnöistä sekä datan prosessoinnista. HTTP (*engl. Hyper Text Transfer Protocol*) on yleisin tiedonvälitykseen palvelimen ja asiakasohjelman välillä käytetty protokolla. Kuvassa 2 on esitetty yksinkertainen asiakasohjelma-palvelin arkkitehtuuri. [43]

Yksi palvelimen rajapinnan arkkitehtuurityyli on REST (*engl. Represantional State Transfer*) API, jota noudattamalla tämänkin työn verkkopohjaisen sovelluksen palvelimen rajapinta tullaan toteuttamaan. REST on arkkitehtuurinen tyyli, joka määrittelee rajoitteita verkkopalvelun luomiselle. Yksi näistä rajoitteista on esimerkiksi tilattomuus: palvelin ei pidä minkäänlaista tilaa yllä vaan se on asiakasohjelman tehtävä [24, s. 1, 3]. Yleensä REST APIn rajapintaa käytetään HTTP-protokollan käskyillä: GET – pyydä dataa tai resurssia, POST – lähetä tai luo dataa tai resurssia, PUT – päivitä dataa tai resurssia ja DELETE – poista dataa tai resurssia [44, s. 104].



Kuva 2. Yksinkertainen asiakasohjelma-palvelin arkkitehtuuri

Asiakasohjelma voi esimerkiksi hakea palvelimelta tietoja hyödyntämällä REST API: rajapintaa yksinkertaisella jQuery [44, s.106] pyynnöllä, jossa haetaan tietoja HTTP *GET* pyynnöllä osoitteesta *uri* (joka on määritelty aiemmin APIn osoitteeksi) ja jos palvelimelta saadaan vastaus, se kirjoitetaan konsoliin. Tälläinen esimerkkiohjelma on esitetty ohjelmassa 2.

```
$.ajax({
    type: ’GET’,
    url: uri,
    dataType: ’json’
    success: function (data) {
          console.log(data)
    }
})
```

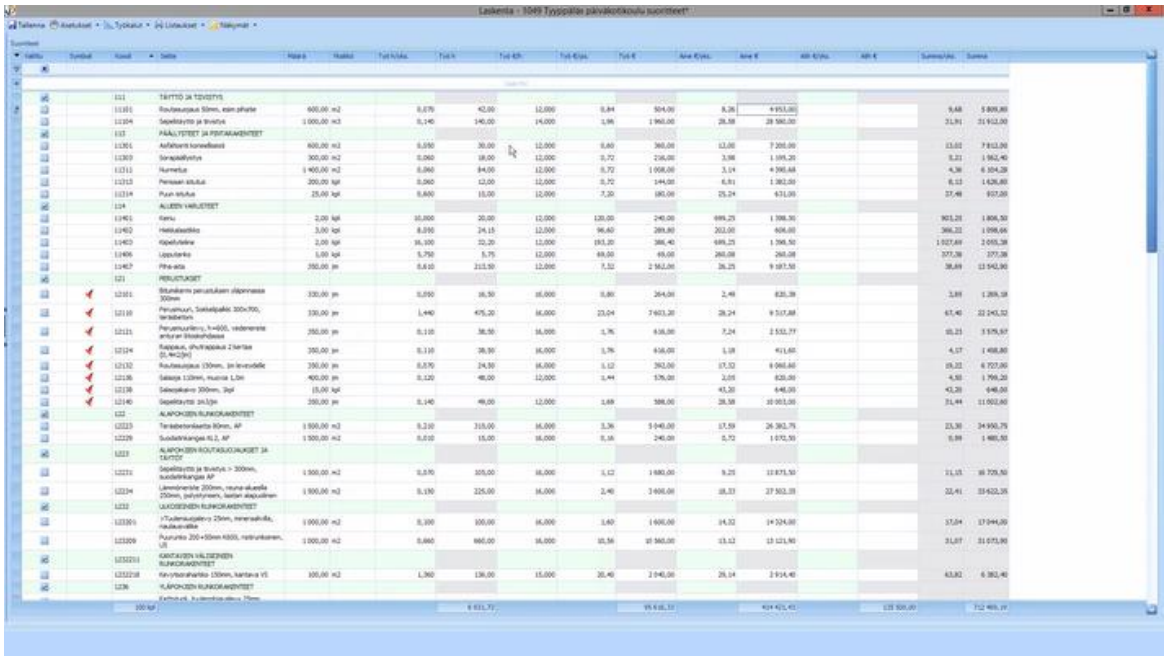
Ohjelma 2. REST API-pyyntö

Palvelimelle voi myös lähettää dataa pyynnön yhteydessä. Palvelinta voi myös käyttää usea asiakasohjelma yhtä aikaa, tai palvelimia, jotka toteuttavat samaa rajapintaa, voi olla useita. [24, s. 3]

2.5 Yleiskuvaus tarjouslaskentasovelluksesta

Toinen suunniteltavista ja toteutettavista työkaluista tulee toimimaan Windowsin työpöydällä kommunikoiden jo olemassa olevan sovelluksen kanssa. Jo olemassa oleva sovellus, jonka kanssa suunniteltavan ja toteutettavan Windowsin työpöydällä toimiva työkalun tulee keskustella, on EVRYn Jydacom Tarjouslaskenta -niminen sovellus [13].

Tarjouslaskentasovellus on Windows Forms -ohjelmointikirjastolla [29] toteutettu Windows-työpöytäsovellus. Oleellisina ominaisuuksina integraation kannalta tarjouslaskentasovelluksessa on *tuoterakenteiden* (esimerkiksi kuinka monta metriä rakennukseen tarvitaan ulkoseinää) ja *suoritteiden* (esimerkiksi kuinka monta ikkunaa rakennukseen tulee) lisääminen ja muokkaaminen. Tiedot haetaan ja tallennetaan tietokantaan. Esimerkki tarjouslaskennan suoritenäkymästä on esitetty kuvassa 19.



Kuva 3. Tarjouslaskennan suoritenäkymä [13]

Kuten kuvasta 3 voidaan nähdä, eri suoritteet näytetään omilla riveillänsä ja ne sisältävät niihin liittyvää tarjouslaskentaa auttavaa tietoa. Tuoterakenteilla on samankaltainen näkymä.

3. OLEMASSA OLEVAT BIM-OHJELMISTOT JA LIITÄNNÄISET

BIM-mallin visualisoimiseen ja käsittelyyn on olemassa paljon erilaisia tapoja. Tämän lisäksi IFC-standardin mukaan määritelty tietomalli voi sisältää jopa miljoonia rivejä standardin mukaista määrittelyä. Ilman hyvin suunniteltua esitystapaa ja standardin mukaista tiedoston jäsentelyä BIM-mallia on vaikea saada visualisoitua käyttäjälle.

Aliluvun 3.1 tarkoituksena on löytää mahdollisimman yhtenäinen ja intuitiivinen esitystapa BIM-mallille katselmoimalla olemassa olevat BIM-ohjelmistot. Katselmoinnissa keskitytään ohjelmiin, jotka ovat ilmaisia ja luotu BIM-mallin tarkastelua varten. Aliluvun 3.2 tarkoituksena on löytää tapa, jolla standardin mukaisesta IFC-tiedostosta saa jäsenneltyä visuaalisen esityksen BIM-mallista käyttäjälle. Tämän työn tarkoituksena ei ole tuottaa standardin mukaista IFC-tiedoston lukemiseen tarkoitettua kirjastoa vaan tässä työssä pyritään etsimään työn tarkoituksiin soveltuva kolmannen osapuolen ohjelmointikirjasto.

3.1 Olemassa olevat BIM-ohjelmistot

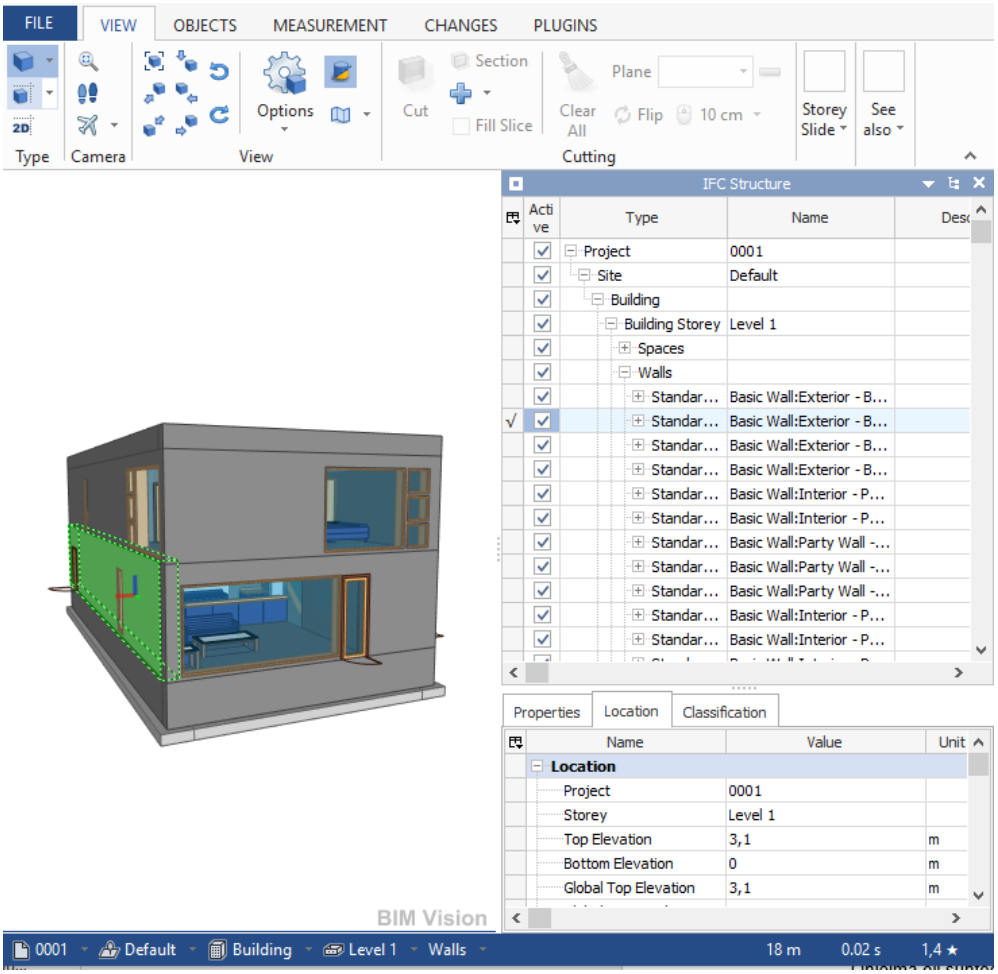
BIMiä tukevia ohjelmistoja on monenlaisia, joista osa on tarkoitettu sen muokkaukseen ja suunnitteluun ja osa pelkästään tarkasteluun. Osa ohjelmistoista käyttää avointa standardia IFC-tiedostoformaatille ja osa käyttää omia natiiveja tiedostomuotoja. Ohjelmistoja on kaupallisia, ilmaisia ja avointa lähdekoodia. Tässä luvussa tarkastellaan kolmea ohjelmistoa, jotka pystyvät avaamaan IFC-tiedostoformaatin ja ovat ilmaisia käyttää. Sovelluksien yleisessä tarkastelussa käytettiin buildingSMARTin tekemää yksinkertaisen talon mallin [10] sisältävää esimerkki IFC-tiedostoa, joka sisälsi noin 38000 riviä ja suorituskyvyn testauksessa kompleksia mallia, joka sisälsi noin 6 miljoonaa riviä.

3.1.1 BIM Vision

BIM Vision [12] on Datacompin kehittämä BIMin tarkasteluun tarkoitettu työpöytäsovellus. Sovelluksen käyttöliittymä on esitetty kuvassa 4. Sovelluksessa IFC-tiedostosta ladatun ja jäsenneltyn BIM-mallin geometriaa pystyy tarkastelemaan sekä kaksi- että kolmiulotteisesti. Kaksiulotteisessa mallissa käyttäjä pystyy etenemään rakennuksen kerros kerrallaan ja tarkastelemaan pohjapiirrosmaisia kuvia rakennuksesta.

Mallista saatu objektihierarkia näytetään puurakenteisessa / kansiomaisessa listassa. Listasta valitut objektit korostetaan kolmiulotteisessa mallissa. IFC-tiedostosta saadun hierarkian lisäksi ohjelma lajittelee rakenneosat tyypeittäin eli esimerkiksi kaikki tasossa 1 olevat seinät ovat ”seinät”-kansion alla. Jokaisesta rakenneosasta saadaan näkymään osan yleiset ja geometriset tiedot valitsemalla osa. Sovellus osaa myös yhdistää IFC-tiedostosta saadun rakenneosien materiaalitiedon niihin kuuluviin osiin.

Ohjelma on hyvin nopea lukemaan IFC-tiedostoja ja generoimaan visuaalisen esityksen tiedoston sisällöstä. Testikoneella 6 miljoonaa rivisessä tiedostossa kesti noin 35 sekuntia jäsennellä IFC-tiedosto ja generoida kolmiulotteinen malli sen sisältämän geometrisen tiedon pohjalta. Sovellukseen pystyy lisäämään erilaisia liitännäisiä, jotka mahdollistavat esimerkiksi BIM-mallin validoinnin ja testauksen. Sovelluksessa pystyy myös tallentamaan malleja sovelluksen omana natiivitiedostona, jotta tiedoston luku nopeutuu. Sovelluksessa pystyy myös laskemaan eri rakenneosien etäisyyksiä toisistaan ja osien pinta-aloja. Sovelluksessa ei pysty muokkaamaan mallia millään tavalla.



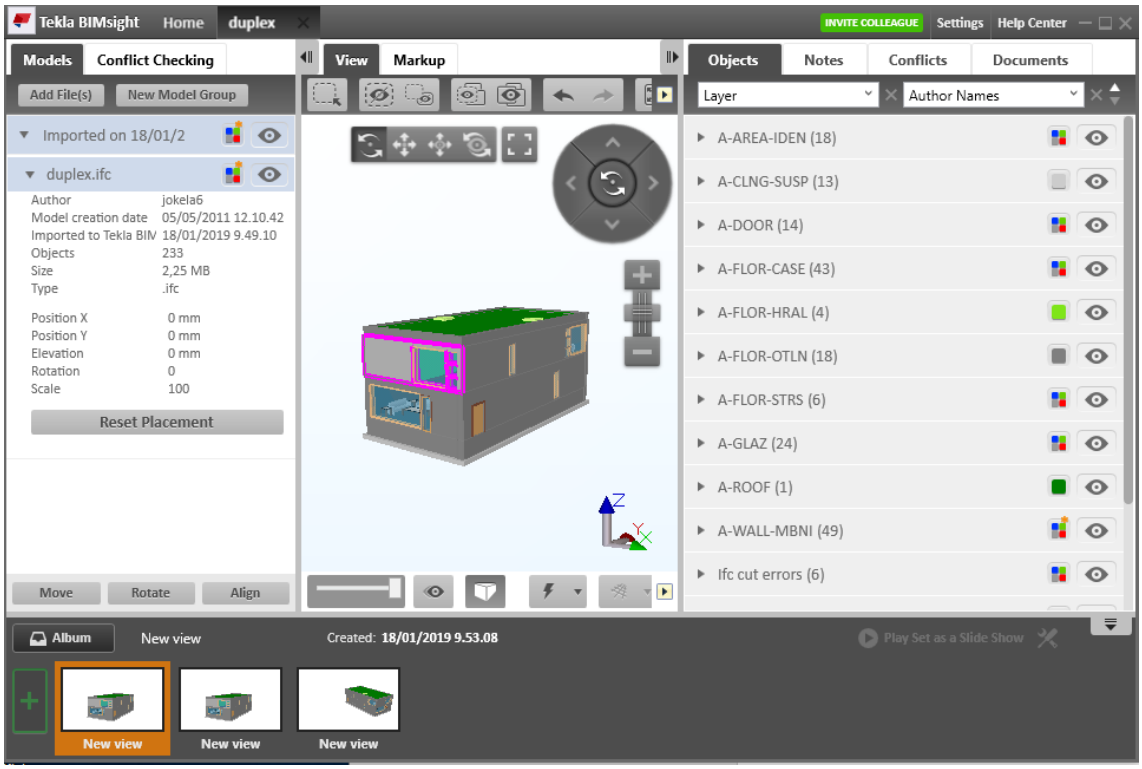
Kuva 4. BIM Vision

3.1.2 Tekla BIMsight

Tekla BIMsight [37] on Teklan kehittämä BIMin tarkasteluun ja projektiin osallistujien yhteistyöhön kehitetty ilmainen työpöytäsovellus. Sovelluksen käyttöliittymä on esitetty kuvassa 5. Ladattua mallia voi tarkastella vain kolmiulotteisesti. Kolmiulotteista mallia pystyy pyörittelemään ja sen osia pystyy korostamaan hiirellä.

Sovelluksessa eri objektit näytetään listana, jota voi suodattaa tai järjestää haluamallaan tavalla. Järjestäminen onnistuu esimerkiksi tyypin, nimen tai kerroksen mukaan. Sovelluksessa pystyy mallin päälle piirtämään ja laittamaan eri osille muistiinpanoja. Objekteja voi myös uudelleen värittää ja piilottaa. Ohjelma osaa myös tarkastaa mallin rakenneosien törmäysten tai konfliktien varalta. Ohjelma osaa myös yhdistää malleja.

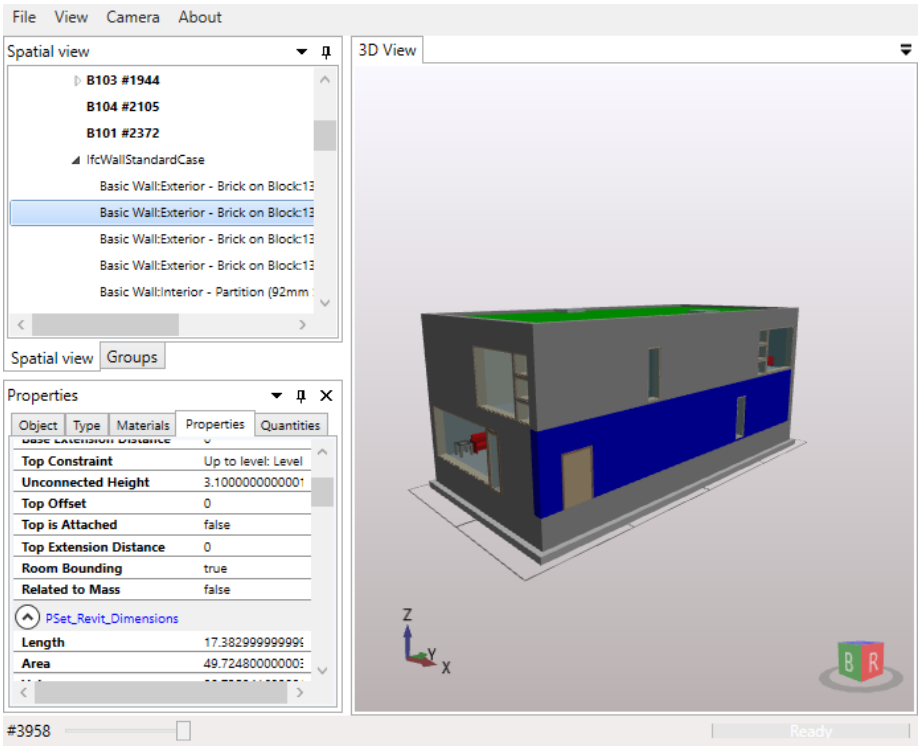
Ohjelma oli hidas IFC-tiedoston avaamisessa verrattuna muihin sovelluksiin. Testikoneella kesti avata 6 miljoonaa riviä pitkä IFC-tiedosto näytettäväksi noin 18 minuuttia. Sovelluksesta pystyy tallentamaan muokatun tiedoston sovelluksen natiiviksi binääritiedostoksi nopeampaa käsittelyä varten.



Kuva 5. Tekla BIMsight

3.1.3 XbimXplorer

XbimXplorer [40] on xBimTeamin kehittämä työpöytäsovellus, joka perustuu saman ryhmän kehittämään avoimen lähdekoodin BIM-ohjelmointikirjastoon. Käyttöliittymä on esitetty kuvassa 6. Sovelluksessa pystyy tarkastelemaan mallia kolmiulotteisesti ja pyörittelemään mallia vapaasti. Osia pystyy valitsemaan ja korostamaan mallista.



Kuva 6. XbimXplorer

Sovelluksessa näkee IFC-tiedostosta ladatun rakenneosahierarkian kansiomaisessa rakenteessa ja sovellus lajittelee objektit myös tyypin mukaisesti. Sovelluksessa pystyy myös validoimaan IFC-tiedoston oikeellisuuden. Sovellus sisältää kehittäjän ikkunan, jossa esimerkiksi osan valintakäskyjä voi antaa komentoriviltä

Sovellus on suhteellisen hidas IFC-tiedoston avaamisessa verrattuna muihin ohjelmiin. Testikoneella kesti noin 6 minuuttia avata 6 miljoonaa riviä pitkä IFC-tiedosto näytettäväksi. Sovelluksessa ei pysty muokkaamaan mallia, mutta sen pystyy muuntamaan joko sovelluksen natiiviksi tiedostoksi tai IFC-tiedostoksi.

3.1.4 Yhteenveto

Yleisesti BIM-tarkastelusovelluksissa objektihierarkiaa pystyi tarkastelemaan puu/kansiorakenteessa. IFC-tiedostosta jäsennellyn ja generoidun kolmiulotteisen mallin näki vapaasti pyöriteltävänä. Kaikissa sovelluksissa oli aputoimintona kolmiulotteisesta mallista osan korostus, joka korostui myös kansiohierarkiassa samanaikaisesti ja sen tietoja pystyi tarkastelemaan. Hyvin suuren tiedoston jäsentelynopeus ja kolmiulotteisen mallin generointi geometriatiedosta vaihteli hyvin paljon, mutta yleensä se oli useita minuutteja. Käyttöliittymä sisälsi jokaisessa sovelluksessa ainakin kolmiulotteisen mallin, rakenneosat ja rakenneosien tiedot. Yleisenä testausominaisuutena oli avatun mallin validointi.

3.2 BIM-liitännäiset

Työssä suunniteltaviin ja toteutettaviin BIM-sovelluksiin on järkevä käyttää kolmannen osapuolen liitännäisiä kompleksin IFC-tiedoston jäsentelyn ja siitä generoidun kolmiulotteisen mallin näyttämistä varten. Kolmannen osapuolen kirjastolle funktionaaliset vaatimukset ovat IFC-tiedoston jäsentely ja kolmiulotteisen mallin näyttäminen käyttöliittymässä. Kolmannen osapuolen liitännäistä täytyy myös pystyä käyttämään web-kehityksessä. Tässä aliluvussa vertaillaan ja arvioidaan yhtä työpöytäsovelluksen liitännäisille tarkoitettua rajapintaa ja kahta avoimen lähdekoodin kirjastoa.

3.2.1 BIM Vision API

Luvussa 3.1.1 tarkasteltuun BIM Vision [12] -sovellukseen on mahdollista vapaasti luoda kaupallisia liitännäisiä sille luodun ohjelmointirajapinnan kautta. Ohjelmointirajapinnan kautta pystyy lisäämään omaa toiminnallisuutta kuvassa 4 esitettyyn käyttöliittymään. Ohjelmointirajapintaa voi käyttää ohjelmointikielillä *C++* ja *C#*. Rajapinta tarjoaa työkalut objektihierarkian tarkastelulle ja sillä pystyy visualisoimaan kolmiulotteisen mallin ja valinnan. BIM Vision API ei kuitenkaan tarjoa työkaluja web-sovelluksen kehittämiseen. Yksinkertaisen napin, jolla pystyy valitsemaan objektin voi luoda helposti APIa käyttäen:

```
api = new ApiWrapper(pid);
button1 = api.CreateButton(0, buttonClick);
api.SetButtonText(button1, "Select object", "Test");
api.Select(id, true);
```

Ylläolevassa ohjelman palassa ensin luodaan API, jonka jälkeen määritellään liitännäiseen uusi nappi, joka valitsee muuttujassa *id* sijaitsevan objektin tunnisteen perusteella objektin.

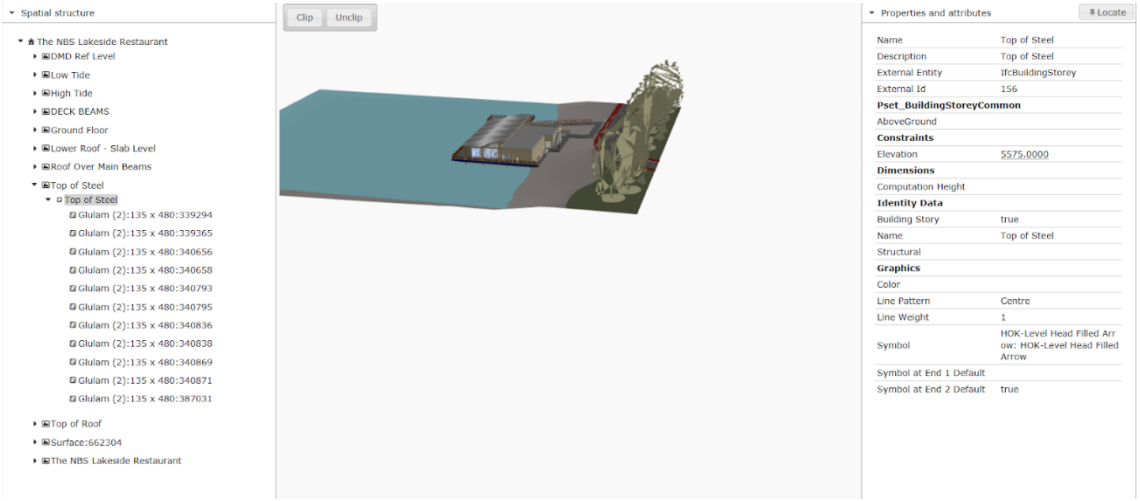
3.2.2 xBIM

xBIM [41] on luvussa 3.1.2 esitellyn XbimXplorerin kehittämiseen käytetty avoimen lähdekoodin kirjasto. xBIM:in käyttö on CDDL Open Source -lisenssin alainen eli sitä saa käyttää vapaasti osana laajempaa kokonaisuutta, kunhan julkaisee lisenssin alaiseen lähdekoodiin tehdyt muutokset samalla lisenssillä avoimeksi lähdekoodiksi [38]. xBIM-kirjastoa päivitetään säännöllisesti

ja sillä on laaja dokumentaatio. XbimXplorer on myös avointa lähdekoodia käyttäen kirjastoa, joten samanlaisen sovelluksen luominen ja muokkaaminen on suhteellisen helppoa.

xBIM-kirjasto tarjoaa työkalut Windows työpöytäsovelluksien kehittäjille lukea, tehdä ja näyttää IFC-formaatissa olevia BIM-malleja. Kirjastokokonaisuuteen kuuluva xBIM essentials on C#-kielellä kirjoitettu perusta, jota käyttämällä IFC-tiedostoja voi lukea, kirjoittaa, validoida ja integroida. xBIM geometry -kirjasto antaa mahdollisuuden visualisoida ja luoda geometriaa xBIM essentials -kirjaston kanssa. [39]

xBIM tarjoaa myös kirjaston web-sovelluksen kehittämiselle nimeltä XbimWebUI. XbimWebUI on tarkoitettu toimimaan verkkosovelluksen asiakasohjelmana ja sen kanssa täytyy käyttää palvelimen puolella xBIM essentials -ja xBIM geometry -kirjastoja IFC-tiedoston jäsentelyä ja geometrian luomista varten. XbimWebUI käyttää WebGL moottoria kolmiulotteisen kuvan renderöimiseen ja käsittelyyn [42]. Tällä tekniikalla luotua kolmiulotteista mallia pystyy vapaasti pyörittelemään ja tarkastelemaan. xBIM Teamin tällä teknologialla luoma esimerkki kolmiulotteisesta mallista ja objektihierarkiasta verkkopohjaisena ratkaisuna on esitetty kuvassa 7.



Kuva 7. XbimWebUI esimerkki

xBIM-ohjelmointikirjasto on suhteellisen suoraviivainen käyttää. Esimerkiksi kuvan 8 mukaisen buildingSMARTin luoman esimerkkitalon [10] voi saada yksinkertaisesti generoitua IFC-tiedostosta lisäämällä *XbimWebUI* komponentin *xViewer* asiakasohjelmaan JavaScript koodina:

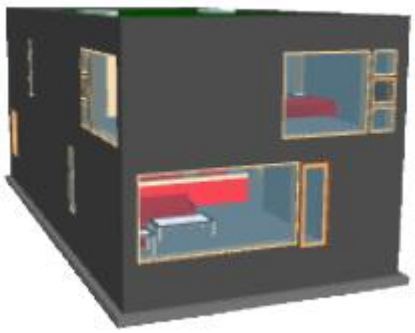
```
var viewer = new xViewer('viewer');
viewer.load('test.wexbim');
viewer.start();
```

ja lisäämällä palvelimelle ohjelman 3 mukaisen IFC-tiedoston konvertterin.

```
using (var m = IfcStore.Open("testi.ifc")) {
   var context = new Xbim3DModelContext(m);
   context.CreateContext();
   var wexBimFilename = Path.ChangeExtension("testi.ifc", "wexbim");
   using (var wexBiMfile = File.Create(wexBimFilename)) {
       using (var wexBimBinaryWriter = new BinaryWriter(wexBiMfile)) {
          m.SaveAsWexBim(wexBimBinaryWriter);
             wexBimBinaryWriter.Close();
       }
       wexBiMfile.Close();
   }
}
```

Ohjelma 3. palvelimen puolen IFC-tiedoston konvertteri [39]

Ohjelman 3 alussa avataan ja jäsennellään välimuistiin tiedosto *testi.ifc*, joka sisältää IFC-muotoista dataa. Tämän jälkeen jäsennellyn tiedoston perusteella luodaan kolmiulotteinen geometria, joka tallennetaan XbimWebUI:n ymmärtämään muotoon eli wexbim-binääritiedostoformaattiin. Luotu wexbim-tiedosto asiakasohjelma voi sitten hakea palvelimelta esimerkiksi luvussa 2.4 esitellyn web API-arkkitehtuurin avulla. XbimWebUI voi sitten avata suoraan tämän sille natiivin ja optimoidun wexbim-binääritiedoston.



Kuva 8. xBIM esimerkki

xBIM-ohjelmointikirjasto täyttää funktionaaliset vaatimukset eli kolmiulotteisen mallin renderöinnin ja IFC-tiedoston jäsentelyn. xBIM-ohjelmointikirjastoa voi myös käyttää apuna web-sovelluksen kehityksessä.

3.2.3 BIMServer

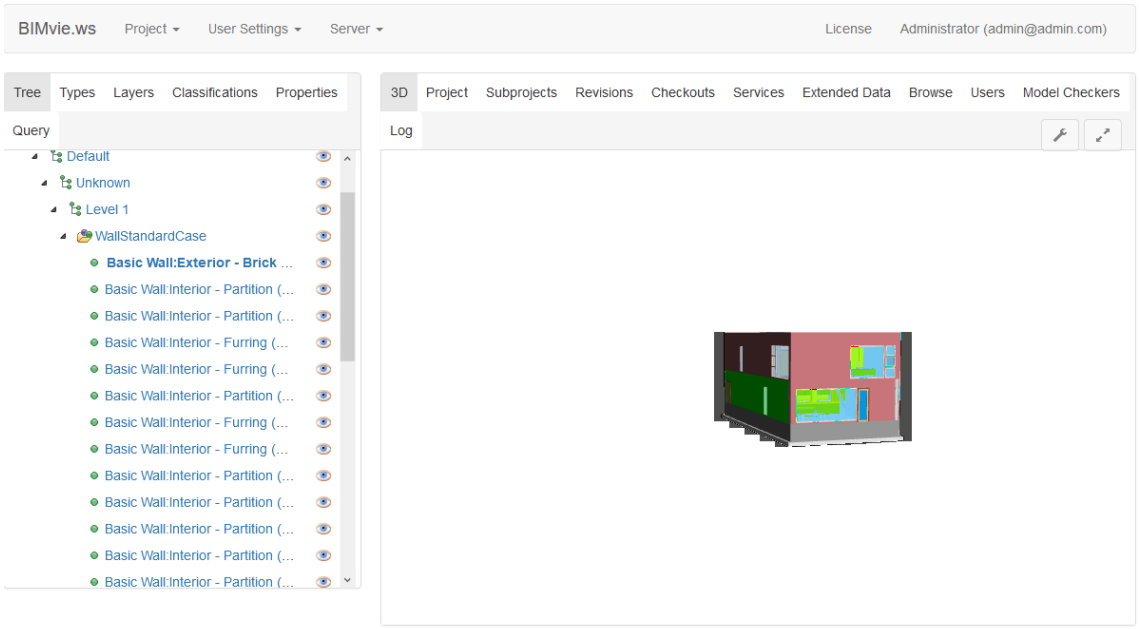
BIMServer [6] on BIMServer.org:in kehittämä avoimen lähdekoodin itsenäinen palvelinohjelmisto, joka antaa mahdollisuuden säilyttää ja hallita IFC-tiedostoja. BIMserveriä saa käyttää GNU Affero General Public Licencen alaisena. Tämän lisenssin ehtona on, että ohjelmistoa saa käyttää vapaasti, mutta sen muokkaukset ja siihen liitetty laajempi kokonaisuus on julkaistava saman lisenssin alaisena avoimeksi lähdekoodiksi [14]. BIMserver on tehty Java-ohjelmointikielellä. BIMServer tukee natiivisti IFC-tiedostoja ja on rakennettu niiden ympärille, joten tiedostoa ei tarvitse muuntaa ennen käyttöä. BIMServerin pohjimmaisena ideana on, että siihen voi tehdä asiakasohjelma-liitännäisiä, jotka tukevat omia käyttötarkoituksia. BIMserveriä päivitetään säännöllisesti, mutta sillä on keskeneräinen dokumentaatio.

BIMserveriä voi käyttää esimerkiksi BIMvie.ws-liitännäisellä [32], jonka tuottama lopputulos buildingSMARTin esimerkkitiedostolla on esitetty kuvassa 9. Liitännäisiä on myös mahdollista luoda itse käyttäen BIMserverin JavaScript-ohjelmointirajapintaa. BIMserveristä voi hakea ja logittaa kaikki projektit esimerkiksi käynnistämällä BIMserver-palvelinohjelman ja kutsumalla sitä asiakasohjelmassa seuraavasti:

```
var api = new BimServerClient("../..");
api.init((client, version) => {
    console.log(version.version);
});
api.call("Bimsie1ServiceInterface", "getAllProjects", {onlyTopLevel: true,
onlyActive: true}, function(data){
          data.forEach(function(project){
                console.log(project);
          });
    });
```

BIMserver APIsta voi hakea esimerkiksi projekteja ja niiden tietoja. BIMserver on itsenäinen palvelin ja sitä ei voi liittää osaksi laajempaa kokonaisuutta, ellei halua julkaista koko työtä avoimeksi

lähdekoodiksi samalla lisenssillä. BIMserverin avulla pystyy jäsentelemään IFC-tiedoston ja renderöimään kolmiulotteisen mallin IFC-tiedostosta.



Kuva 9. BIMserver ja BIMvie.ws

3.2.4 Vertailu ja yhteenveto

Kaikki edellä mainitut liitännäiset ovat hyvin erilaisia ja ne ovat luotu eri tarkoituksiin. Kolmen edellä esitellyn kirjaston ominaisuudet on esitetty vierekkäin taulukossa 1.

	BIM Vision	xBIM	BIMserver
Tyyppi	Työpöytäsovelluksen ohjelmointirajapinta	Ohjelmointikirjasto	Itsenäinen palvelinohjelmisto
Ohjelmointikieli	C++ / C#	C++ + C# + JavaScript	Java + JavaScript
Lisenssi/oikeudet	Tekijänoikeudet kuuluvat liitännäisen kehittäjälle kehittäjälle	CDDL Open Source	GNU Affero General Public Licence
Open source	Ei	Kyllä	Kyllä
Web-kirjasto	Ei	Kyllä	Kyllä
Windows työpöytäsovellus -tuki	Kyllä	Kyllä	Ei
IFC-tuki	Kyllä	Kyllä	Kyllä
3D-Mallin renderöinti	Kyllä	Kyllä	Kyllä
Objektihierarkia	Kyllä	Kyllä	Kyllä

Taulukko 1. BIM-liitännäisten vertailu

Kuten taulukosta voidaan nähdä, xBIM ja BIMserver täyttää vaatimukset työssä suunniteltavalle BIM web -työkalulle. Työssä on kuitenkin tarkoitus integroida tuotettu käsittelytyökalu osaksi portaalia ja jo olemassa olevaa palvelinohjelmistoa, jonka vuoksi BIMserverin lisenssi ja eri toteutuskieli voi tuottaa ongelmia. xBIMin lisenssi ei rajoita ohjelmointikirjaston rajapinnan käyttöä ja sen ohjelmointikirjastomaisen arkkitehtuurin vuoksi se on mahdollista integroida jo olemassa olevaan infrastruktuuriin. xBIM on myös liitännäisistä ainut, joka tukee sekä web-kehitystä että Windows työpöytä -kehitystä. Näiden tietojen valossa xBIM on luonnollinen vaihtoehto liitännäiseksi BIM-mallin näyttämiselle ja jäsentelylle.

4. BIM-TARKASTELUTYÖKALUN SUUNNITTELU

Kuten johdannossa mainittiin, tässä työssä suunnitellaan BIM-työkalu, joka mahdollistaa tietojen siirron BIM-mallista EVRYn Jydacom tarjouslaskentasovellukseen. Ensimmäinen vaihe BIM-lisäohjelman suunnittelussa on suunnitella intuitiivinen BIM-mallin tarkasteluun ja tietojen lajitteluun tarkoitettu sovellus, jossa on selkeä rajapinta lajiteltujen tietojen hakemiselle muihin sovelluksiin. Vaikka BIM-työkalu tullaan liittämään ensisijaisesti vain tarjouslaskentasovellukseen, itsenäisen tarkastelusovelluksen suunnittelulla saavutetaan riippumattomuus tarjouslaskentasovelluksesta. Tämä mahdollistaa tarkastelutyökalun itsenäisen kehityksen, sekä mahdollistaa toteutetun BIM-työkalun liittämisen muihin toiminnanohjausta avustaviin sovelluksiin tarvittaessa ilman suurempia muutoksia.

Alkuperäisenä suunnitelmana oli toteuttaa ratkaisu web-sovelluksena osaksi EVRYn Jiiri web -portaalia, joka pystyy sitten kommunikoimaan työpöydällä toimivan tarjouslaskentasovelluksen kanssa verkon yli. Suunnittelun edetessä BIM-työkalu kuitenkin päätettiin toteuttaa sekä web-sovelluksena että Windows työpöytäsovelluksena. Web-sovellus toimii BIM-mallin tarkastelemisessa sekä *proof-of-conceptina* eli todisteena konseptin toimivuudesta tulevaisuutta varten. Työpöytäsovelluksen tarkoituksena on mahdollistaa sen välitön yhteistyö tarjouslaskentasovelluksen kanssa. Tämän työn kirjoittajan rooli tässä projektissa on tehdä sovelluksista suunnitelma, toteutus ja dokumentaatio. Suunnittelussa painotetaan erityisesti tietojen haun ja lajittelun käyttäjäkokemusta, selkeyttä sekä nopeutta.

Jako web- ja työpöytäsovellukseen päädyttiin tekemään ohjelman saumattoman toimivuuden, käytettävyyden ja tuotteistuksen parantamiseksi tarjouslaskentatyökalun ollessa erillinen tuote web-portaalista. Molempien sovellusten toteutuksen pohjana käytetään samoja tässä luvussa esitettäviä vaatimuksia, toiminnallisia suunnitelmia sekä käyttöliittymäsuunnitelmia. Sovellukset eroavat aliluvuissa 4.4 ja 4.5 esitetyin tavoin arkkitehtuurillisesti, sekä tietenkin käyttävät eri teknologiaa käyttöliittymän toteutuksessa. Käytettävistä teknologioista ja niiden käyttökohteista on kerrottu aliluvussa 4.3. Tämän luvun tarkoituksena on esitellä sovelluksien itsenäisesti toimivat ominaisuudet ja luvussa 5 on tarkoitus kertoa sovellusten toiminta liitännäisinä.

4.1 Vaatimukset

Käyttötapausvaatimukset BIM-työkalulle on selvitetty etukäteen asiakkaan kanssa ja ne on esitetty seuraavassa olevassa luettelossa:

1. tarjouslaskija voi tuoda BIM-malleja BIM-työkaluun.
2. tarjouslaskija voi tarkastella mallia kolmiulotteisena.
3. tarjouslaskija voi tarkastella mallin tietorakennetta.
4. tarjouslaskija voi lajitella mallin objekteja.

Näiden käyttötapausten pohjalta voidaan laatia BIM-työkalulle *user storyt* eli käyttäjätarinat ja niihin liittyvät tehtävät, jotka on esitetty seuraavassa luettelossa tyylillä ”tarjouslaskijana haluan...”:

1. Tuoda BIM-malleja työkaluun.
2. Säilyttää ladatut mallit työkalussa seuraavaa käyttökertaa varten.
3. Nähdä BIM-tiedostosta generoidun talon kolmiulotteisen mallin.
4. Nähdä objektihierarkian.
5. Valita rakenneosan/osia hierarkiasta.
6. Nähdä valitun objektin tiedot.
7. Ryhmitellä objekteja.
8. Suodattaa objekteja.
9. Yhdistää objektien ominaisuuksia.
10. Liikkua web-portaalista BIM-työkaluun.

Vaatimuksista voidaan tämän ja kolmannen osapuolen komponentin valinnan jälkeen aloittaa muodostamaan työkalun tarkempaa toiminnallisuutta ja teknisen toteutuksen arkkitehtuuria. Ohjelmassa on myös tärkeää ottaa huomioon toteutuksen yhtenäisyys olemassa olevien toiminnanohjausohjelmien kanssa sekä ohjelman suorituskyvyn kilpailukyky verrattuna samankaltaisiin ohjelmiin.

4.2 Toiminnallisuus

Tarkempi toiminnallisuus voidaan muodostaa luvussa 4.1 esitettyjen käyttäjätapausten ja käyttäjätarinoiden perusteella. Toiminnallisuus voidaan jaotella viiteen eri kokonaisuuteen: tiedoston lataus, kolmiulotteinen malli, objektihierarkia, objektin ominaisuudet ja mukautettu näkymä.

Tiedoston latauksessa on otettava huomioon yleisimmät tiedostoformaatit sekä käytettävän kolmannen osapuolen liitännäisen, xBIMin, natiivi tiedostoformaatti. Tiedoston lataus muistiin tulee onnistua sekä yksinkertaisen Windowsin natiivin tiedostojenhallinta komponentin avulla käyttäjän aktivoimana että tietokannasta kolmannen osapuolen liitännäisen natiivitiedostoformaatissa. Tiedoston latauksessa tulee tukea yleisimpiä BIM-tiedostoformaatteja, jotka ovat IFC, IFCZip, IFCXml. Lataamisen jälkeen tiedosto prosessoidaan ja tallennetaan tietokantaan kolmannen osapuolen liitännäisen natiivimuodossa seuraavan käynnistyskerran nopeampaa latausta varten. Ohjelman täytyy tukea vanhempaa IFC-mallia IFC3x2 ja uudempaa mallia IFC4. Malleja täytyy pystyä lataamaan tietokantaan useampi valmiiksi ja näytettävää mallia voi vaihtaa työkalussa helposti suoritusaikana.

Ladattu tiedosto tulee näyttää kolmiulotteisena mallina, jota pystyy lähentämään, loitontamaan ja pyörittelemään vapaasti. Mallista täytyy pystyä valitsemaan eri objekteja valitsemalla ne hiirellä, jonka jälkeen ne korostuvat korostusvärillä kolmiulotteisessa mallissa. Objekteja voi korostaa monta yhtäaikaisesti pitämällä näppäimistön ’control’-näppäintä pohjassa. Ilman ’control’-näppäintä edellinen valinta unohdetaan ja korostetaan vain uusi valinta. Kun kolmiulotteisesta mallista korostetaan objekteja, vastaavat objektit korostetaan objektihierarkiassa. Mallista täytyy pystyä myös paljastamaan vain valitut objektit, jolloin muut valittujen objektien ympärillä olevat objektit piilotetaan tai häivytetään. Tästä täytyy pystyä myös palaamaan tilaan, jossa kaikki objektit näytetään.

Objektihierarkiassa näytetään mallista jäsennelty fyysisten objektien hierarkia, josta esimerkki on esitetty kuvassa 1. Objektihierarkiasta voi korostaa monia objekteja yhtäaikaisesti ja käyttää Windowsin resurssienhallinnasta tuttuja pikanäppäimiä hyödyksi. Objektihierarkiasta objekteja korostaessa vastaavat objektit korostetaan myös kolmiulotteisessa mallissa.

Kun kolmiulotteisesta mallista tai objektihierarkiasta korostetaan objekteja, niiden ominaisuudet näytetään erillisessä näkymässä listana. Objektin ominaisuudet ryhmitellään listaan ominaisuuksien BIM-tiedostossa määriteltyjen ominaisuusjoukkojen mukaisesti. Korostaessa useita objekteja mallista tai objektihierarkiasta yhtäaikaisesti, vain viimeisen korostetun objektin ominaisuudet näytetään näkymässä. Jos objektin tietojen hakeminen kestää isosta mallista kauan, tehdään se taustalla ja annetaan latausindikaattori käyttäjälle ’objektin ominaisuudet’-näkymän kohdalla jäädyttämättä käyttöliittymää. Jos objektihierarkiasta on korostettuna usea objekti yhtäaikaisesti, ominaisuudet-näkymässä näytetään vain objektien yhteenlasketut ominaisuudet, esimerkiksi yhteenlaskettu pituus.

Käyttäjän pitää pystyä näkemään vain haluamansa objektit mallista, joten objekteja täytyy pystyä suodattamaan ja lajittelemaan ominaisuuksien, esimerkiksi tyypin, mukaan. Objekteja voi lajitella joko valitsemalla objektin ominaisuudet -näkymästä ominaisuuden tai hakemalla mallissa olemassa olevan ominaisuuden yksinkertaisesta valikosta erillisessä ikkunassa. Kun lajittelu tapahtuu, alkuperäinen objektihierarkia muuttuu mukautetuksi hierarkiaksi, jossa näytetään suodatuskriteerejä vastaavat objektit oikein ryhmiteltynä. Mukautetusta näkymästä pystyy helposti siirtymään takaisin objektihierarkiaan tai jatkamaan lajittelua eri ominaisuuksilla.

4.3 Tarkastelusovelluksessa käytettävät teknologiat

Koska lopullinen sovellus päätettiin jakaa kahteen osaan, käytettäviä eri teknologioita on suhteellisen monta. Verkkopohjaisen sovelluksen kehittämisessä käytetään asiakasohjelmassa TypeScript-ohjelmointikieltä Angular-ohjelmistokehyksen kanssa sekä palvelimella .NET Core -ohjelmistokehystä. Suunniteltavassa työpöytäsovelluksessa käytetään .NET Framework -ohjelmistokehystä ja Windows Forms -ohjelmointikirjastoa.

Kaikki teknologiavalinnat pohjautuvat jo olemassa olevien sovellusten teknologioihin. Suunniteltava verkkopohjainen sovellus liitetään osaksi EVRYn toiminnanohjaus-web-portaalia, joka on toteutettu TypeScriptillä, Angularilla ja .NET Corella. Suunniteltava työpöytäsovellus pitää pystyä liittämään .NET Frameworkilla ja Windows Formsilla toteutettuun työpöytäsovellukseen. Tässä aliluvussa käydään lyhyesti näiden teknologioiden yleisiä piirteitä ja arkkitehtuurimalleja.

4.3.1 TypeScript

JavaScript-ohjelmointikieli on luonnollinen valinta verkkosovelluksen ohjelmointikieleksi, sillä se toimii kaikilla alustoilla ja melkein kaikki modernit selaimet pystyvät suorittamaan sitä. JavaScript on kuitenkin suuremmissa projekteissa vaikeasti ylläpidettävää ja testattavaa dynaamisen tyypityksen vuoksi. Tähän ongelmaan Microsoft on kehittänyt TypeScript-ohjelmointikielen, joka muuntuu JavaScriptiksi käännöksen jälkeen. Typescript lisää staattisen tyypitystason perinteisen JavaScriptin päälle, joka sitten ajetaan kääntäjän lävitse. [30, s. 6]

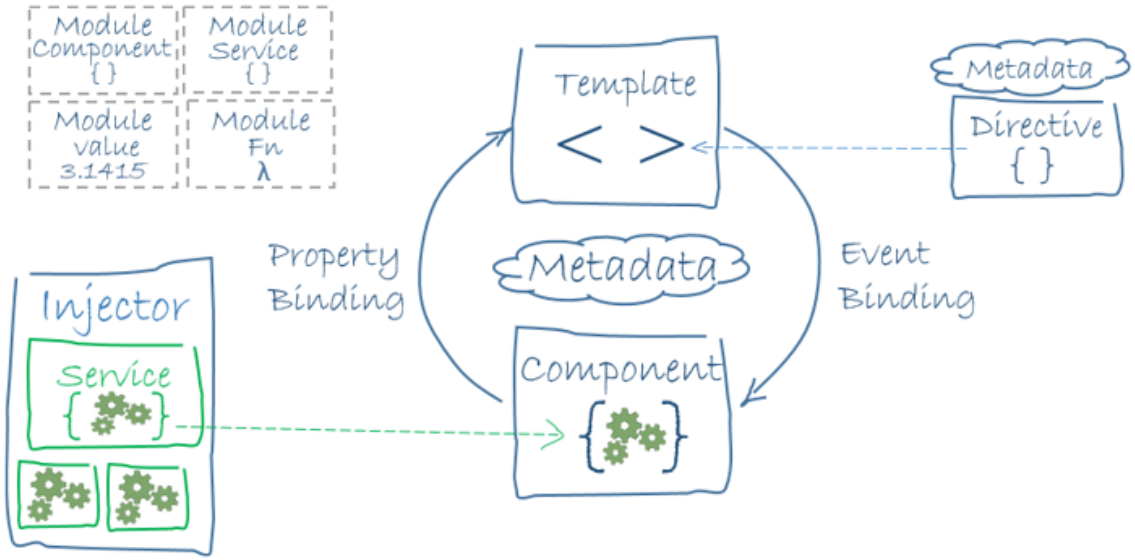
TypeScript toimii kaikissa selaimissa ja sitä pystyy ajamaan Node.js- tai missä tahansa muussa JavaScript-moottorissa, joka tukee ECMAScript 3:a tai uudempaa. Typescriptissä tyypitys on valinnaista, joten jo olemassa olevia JavaScript koodeja/kirjastoja voi käyttää osana TypeScriptiä käyttävää ohjelmaa. TypeScript mahdollistaa verkkokehittäjille staattisen tarkistuksen ja koodin refaktoroinnin, joka tekee ohjelmointikielestä hyvin skaalautuvan. TypeScript-ohjelmointikieli on avointa lähdekoodia. Alla oleva ohjelma on lyhyt esimerkki TypeScriptin muodossa, jossa tulostetaan näytölle perinteisesti ”Hello world!” [30, s. 17].

```
var p = document.createElement('p');
var hello: string = "Hello";
var world: string = "world!";
p.textContent = hello + " " + world;
document.body.appendChild(p);
```

Kuten ohjelmakoodista voidaan nähdä, TypeScript muistuttaa hyvin paljon JavaScriptiä, mutta muuttujat *hello* ja *world* on tyypitetty merkkijonoksi. Jos muuttujalle *hello* olisi antanut esimerkiksi numeron syötteenä ”Hello”:n sijaan, olisi kääntäjä antanut virheilmoituksen. TypeScript mahdollistaa näin ohjelmakoodin staattisen tarkistuksen, jonka vuoksi TypeScript on valittu tämän työn asiakasohjelman ohjelmointikieleksi. [27]

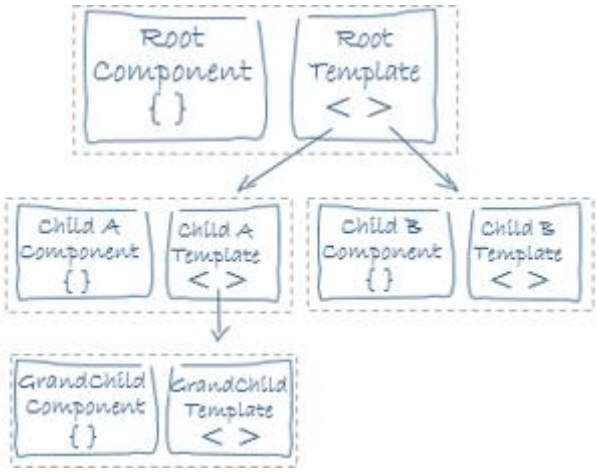
4.3.2 Angular

Yksi vaihtoehto verkkosovelluksessa käytettävän asiakasohjelman toteuttamiseksi on tässä työssä käytettävä Angular-ohjelmistokehys. Angular on Googlen Angular-teamin kehittämä TypeScript-kielellä uudelleenkirjoitettu AngularJS-ohjelmistokehys. Angular on suunniteltu modernien sovellusmaisten verkkosivujen kehittämiseen, jotka toimivat ”*single-page*” periaatteella eli sivua ei tarvitse ladata uudestaan alisivuilla navigoidessa. [15]



Kuva 10. Angular sovelluksen arkkitehtuuri [16]

Angularilla toteutetun sovelluksen arkkitehtuuri voidaan nähdä kuvasta 10. Se yleensä koostuu moduuleista, komponenteista, templateista ja palveluista. Moduulit ovat säiliöitä kaikkien luotujen ohjelman osien esittelylle ja niiden yhteen ryhmittämiselle. Komponentti kontrolloi käyttäjälle näkyvää näkymää sekä sisältää sen logiikan ja datan. Jokaisella komponentilla on template eli malli, joka kertoo Angularille miten näkymä renderöidään. Template on HTML (*engl. Hypertext Markup Language*) -muotoista. Template ja komponentti voivat vaihtaa tietoa keskenään kaksisuuntaisen tietojensitomisen avulla, jossa komponentissa käytetyt muuttujien arvot ovat sidottuja templaten muuttujiin ja templatessa tapahtuvat tapahtumat sidottu komponentin funktioihin. Näkymät, jotka pitävät sisällään komponentin ja templaten, muodostavat yleisesti hierarkkisen mallin, jossa on juurinäkymä, joka pitää sisällään pienempiä näkymiä. Tämä mahdollistaa näkymien vaihtamisen ja piilottamisen dynaamisesti. Tyypillinen näkymähierarkia on esitetty kuvassa 11. [17]



Kuva 11. Angularin näkymähierarkia [16]

Näkymien ja moduulien lisäksi Angular-sovelluksen arkkitehtuuriin kuuluu yleensä yksi tai useampi palvelu eli *service*. Palveluiden tehtävä on laajentaa komponentin ominaisuuksia käyttömukavuuden ja näkymän ulkopuolelle, esimerkiksi hakemalla dataa komponentille palvelimen API:lta tai kirjaamalla virheviestejä verkkoselaimen konsoliin. Palvelulla on ideaalisti yksi vastuualue, jonka ne toteuttavat hyvin. Palvelu toimitetaan komponentille *dependency injection* -suunnittelumallilla, joka mahdollistaa saman palvelun instanssin käytön useassa eri komponentissa. [18]

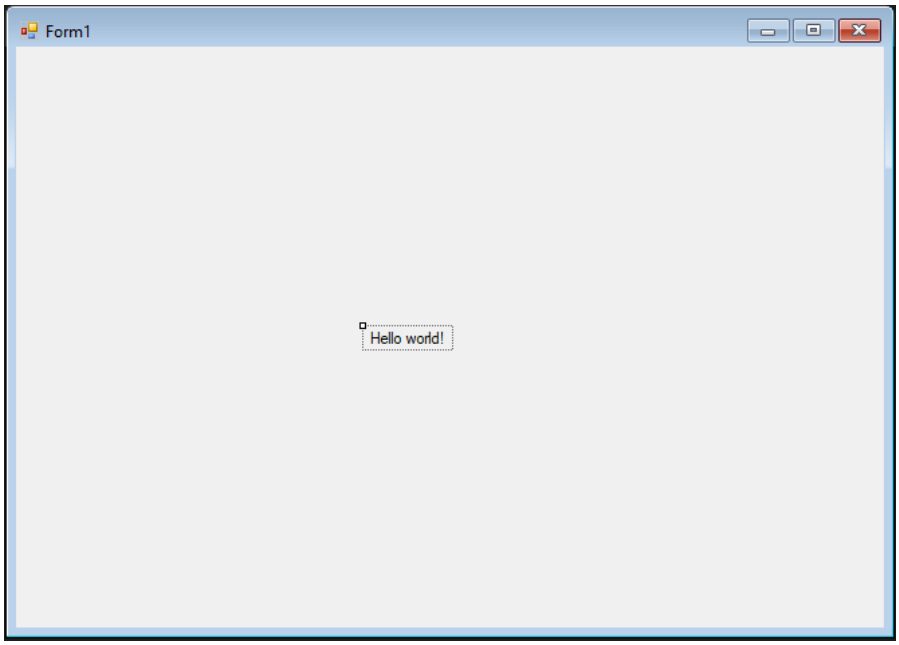
4.3.3 .NET-Ohjelmistokehys

Palvelimen puolella sekä Windows työpöytäsovelluksessa tässä työssä käytetään .NET -ohjelmistokehystä. Web-sovelluksen palvelimella käytetään .NET Core osaa .NET-ohjelmistokehyksestä ja sen käytettynä ohjelmointikielenä on C#. Windows työpöytäsovelluksessa käytetään .NET Framework osaa .NET-ohjelmistokehyksestä ja sen ohjelmointikielenä käytetään Visual Basic -ohjelmointikieltä.

.NET on Microsoftin kehittämä järjestelmäriippumaton avoimen lähdekoodin ohjelmistokehys. Sillä voi kehittää verkko-, puhelin-, työpöytä-, peli- tai esineiden internet-sovelluksia. Sen kanssa voi käyttää C#, F# tai Visual Basic-ohjelmointikieliä. .NET tarjoaa automaattisen muistinhallinnan varaten ja vapauttaen muistia automaattisesti, jotta kehittäjän ei tarvitse huolehtia siitä. Tämän työn verkkosovelluksessa ohjelmointikielenä käytetty C# on Microsoftin suunnittelema moderni olioperustainen ja tyypitysturvallinen ohjelmointikieli. [28]

4.3.4 Windows forms -kirjasto

Windows Forms on Microsoftin kehittämä käyttöliittymäkomponenttikirjasto. Windows Forms pohjautuu käyttöliittymällä näytettäviin lomakkeisiin ja niihin liitettäviin ohjauskomponentteihin. Windows Forms käyttää sen perustana .NET Framework -ohjelmistokehystä. Windows Forms mahdollistaa kehityksen myös graafisen käyttöliittymän kautta. Kuvassa 12 on esitetty Windows Formsilla toteutettu perinteisen ’Hello world’ -sovelluksen graafinen käyttöliittymä. Kuvasta voi nähdä graafisen kehitysympäristön tyylin ja Windows Formsilla toteutetun sovelluksen yleisilmeen. [29]



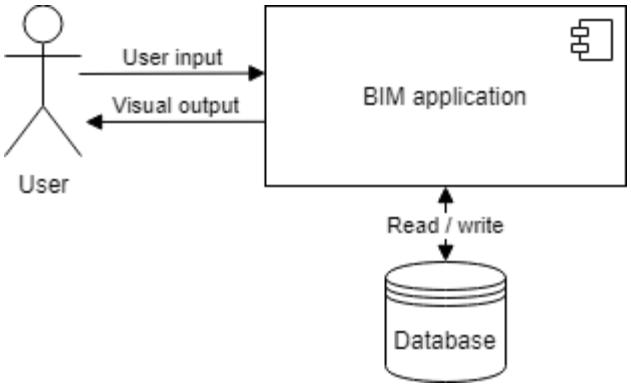
Kuva 12. Windows Formsilla toteutettu “Hello world” -sovellus

4.4 Web-tarkastelusovelluksen arkkitehtuuri

Ensimmäisenä on tärkeää suunnitella web-sovelluksen arkkitehtuuri, sillä se on huomattavasti kompleksisempi ja luo rajoitteita työpöytäsovelluksen arkkitehtuuriin, jos näistä halutaan mahdollisimman yhteneväiset. On tärkeää suunnitella web-sovelluksen arkkitehtuuri niin, että sitä pystyy hyödyntämään myös työpöytäsovelluksen suunnittelussa ja niin että näissä kahdessa sovelluksessa olisi mahdollisimman paljon samoja komponentteja.

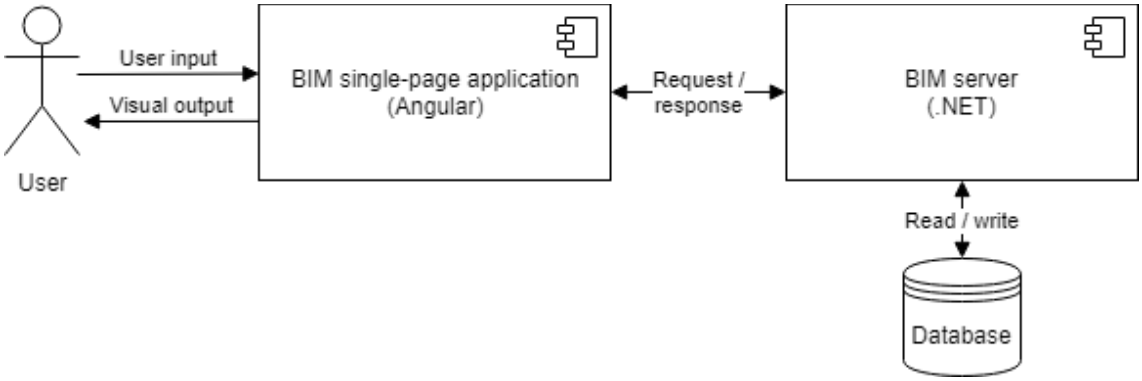
Tavoitteena on tuottaa sovellus, joka toimii vaatimusten mukaisesti ja pystyy myös tarvittaessa tallentamaan tietoa tietokantaan tai lukemaan tietoa tietokannasta. Kolmannen osapuolen BIM-

liitännäiseksi valikoitui xBIM, joka on toteutettu JavaScriptillä ja .NET Frameworkilla. Järjestelmän kontekstikaavio on esitetty kuvassa 13.



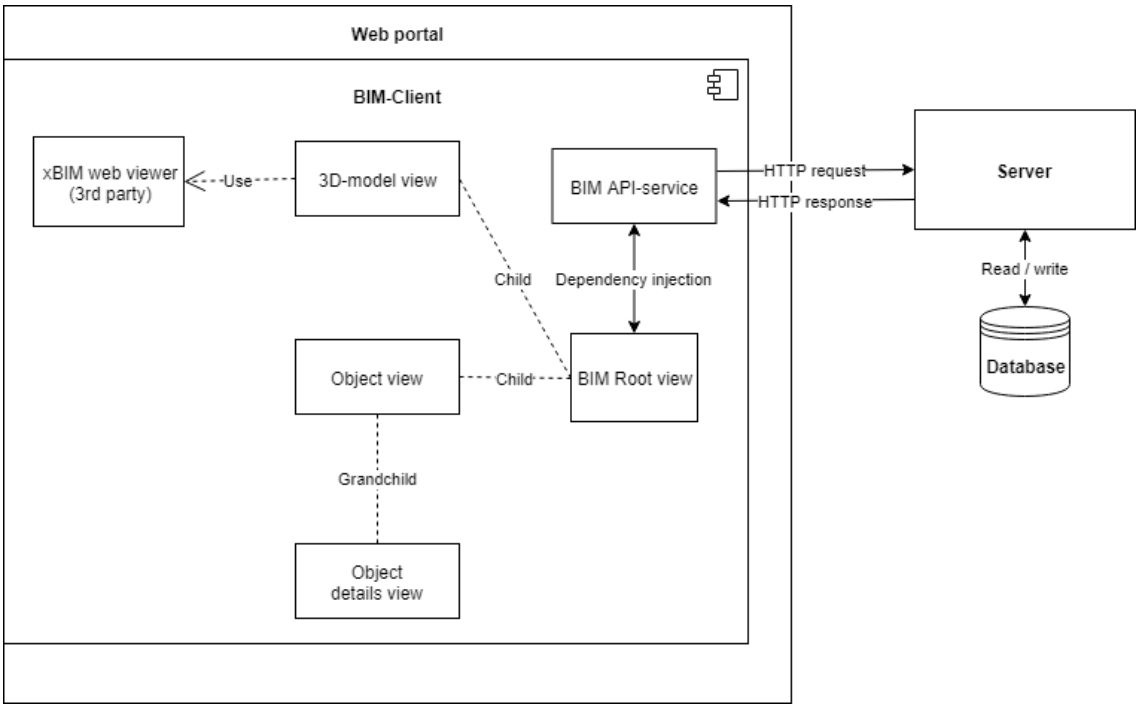
Kuva 13. BIM-työkalun järjestelmäkontekstikaavio

BIM-työkalu on tarkoitus liittää osaksi jo olemassa olevaa toiminnanohjausportaalia omaksi moduulikseen, joten se on tärkeä suunnitella toimimaan samoilla periaatteilla kuin toiminnanohjausportaali. Ohjelma suunnitellaan omaksi kokonaisuudekseen, jonka pystyy liittämään osaksi kokonaisuutta tai irroittamaan ilman suurempia ongelmia tai muutoksia jo olemassa olevaan järjestelmään. Toiminnanohjausportaali käyttää Angular-ohjelmistokehystä asiakasohjelman toteutuksessa ja .NET Core 2 -ohjelmistokehystä palvelimen toteutuksessa, joten ohjelma suunnitellaan käyttämällä näiden periaatteita. Järjestelmän säiliökaavio on esitetty kuvassa 14, jossa BIM-työkalun osat on esitetty osaksi toiminnanohjausjärjestelmää liitettävinä komponentteina.



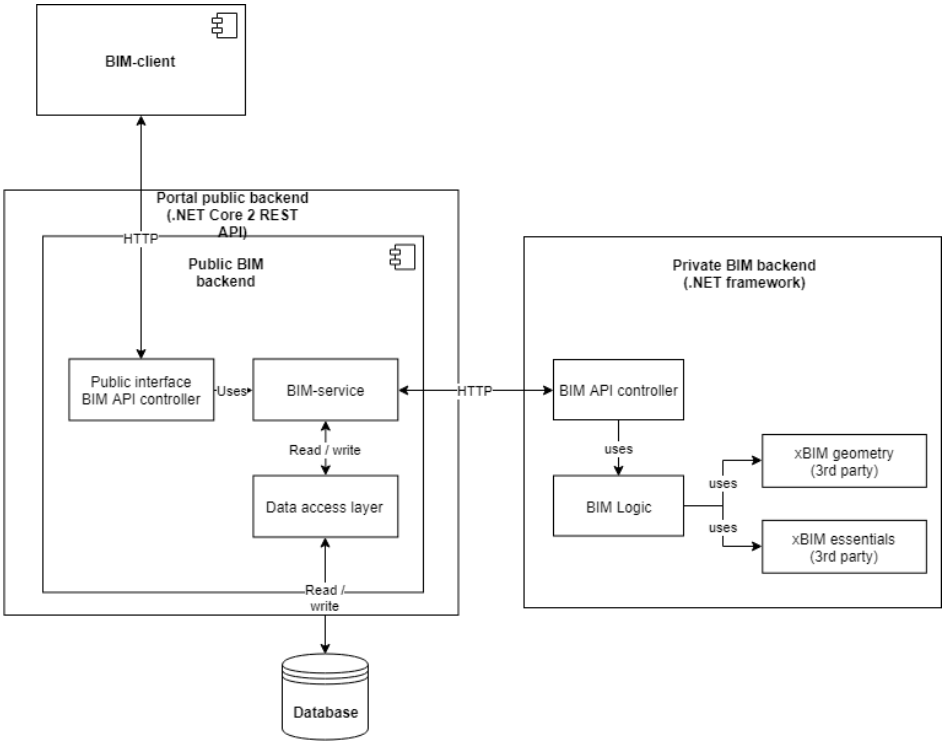
Kuva 14. BIM-työkalun säiliökaavio.

Portaalin liitettävässä BIM-moduulissa täytyy näyttää kolmiulotteinen malli talosta, rakenneobjektien hierarkia ja niiden ominaisuudet. Käyttäjälle näytettävä näkymä koostuu BIM-juurinäkymästä, kolmiulotteinen malli -näkymästä, rakenneobjektinäkymästä ja rakenneosien ominaisuus -näkymästä. BIM-juurinäkymä toimii tiedonvälittäjänä sen lapsinäkymäkomponenttien välillä, sekä käyttää BIM API -palvelua pyyntöjen luomiseen palvelimelle. BIM-asiakasohjelman komponenttikaavio on esitetty kuvassa 15.



Kuva 15. BIM-asiakasohjelman arkkitehtuuri komponenttikaaviona

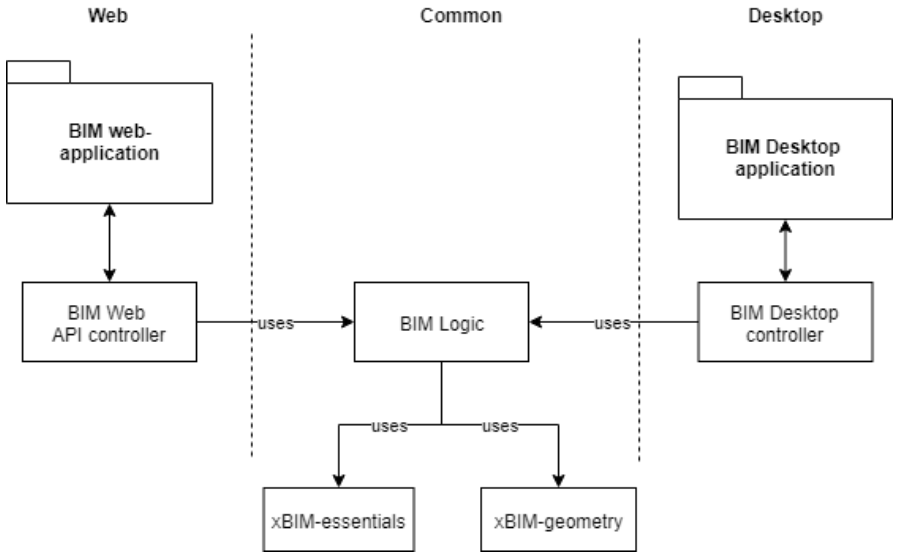
xBIM Essentials- ja xBIM Geometry -ohjelmointikirjastoja käytetään IFC-tiedoston jäsentelyssä ja kolmiulotteisen mallin luomisessa asiakasohjelmaa varten palvelimen puolella. xBIM-geometry -kirjasto toimii vain .NET Framework ympäristössä, joten sitä ei voi liittää osaksi jo olemassa olevaa .NET Core -web-palvelua. Tämän vuoksi täytyy luoda uusi erillinen IFC-tiedoston oikeaan muotoon muuntamiseen keskittyvä palvelu. Autentikointi ja muu portaaliin liittyvä hallinnointi tapahtuu .NET Core -web-palvelussa, joten sen rajapintaa täytyy käyttää http-pyynnöissä. Tämä johtaa siihen, että julkiseen rajapintaan on luotava komponentti, joka lähettää tiedon ja pyynnön eteenpäin toiselle yksityisenä palvelimena toimivalle web-palvelulle. Tästä on kerrottu enemmän luvussa 4.6. Palvelimen arkkitehtuurin komponenttikaavio on esitetty kuvassa 16.



Kuva 16. Palvelimen puolen arkkitehtuuri komponenttikaaviona.

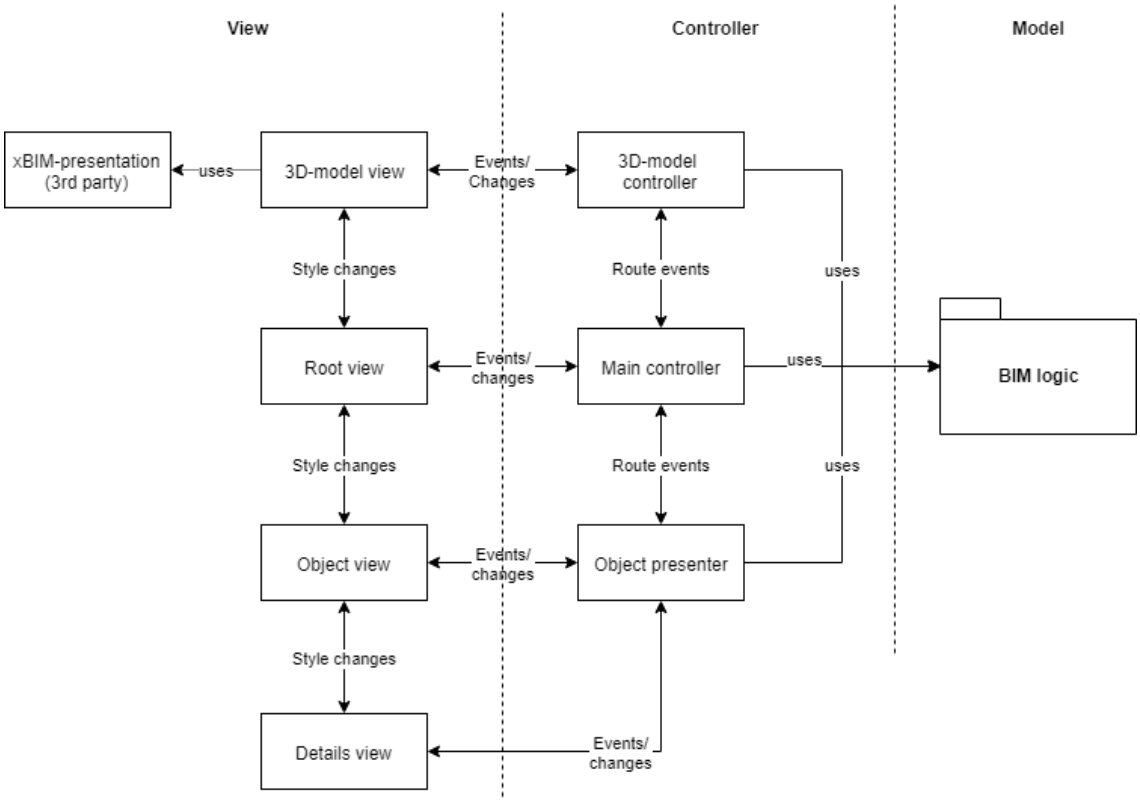
4.5 Työpöydällä toimivan tarkastelusovelluksen arkkitehtuuri

Verkkopohjaisen lisäohjelman palvelin- ja työpöytäsovellus käyttävät .NET -ohjelmointikehystä, joten jos BIM-logiikka toteutetaan kirjastomaiseksi riippumattomaksi logiikaksi, sitä voidaan käyttää molemmissa sovelluksissa. Kuvassa 17 on esitetty havainnollistava komponenttikaavio logiikan yhteiskäytöstä.



Kuva 17. Yhteinen BIM-logiikka

Työpöydällä toimivan lisäohjelman arkkitehtuuri pidetään yhtenäisenä web-sovelluksen arkkitehtuurin kanssa. Työpöydällä toimivan lisäohjelman arkkitehtuuri on esitetty kuvassa 18.



Kuva 18. Windows työpöytäsovelluksen arkkitehtuuri

Kuten kuvasta 18 voidaan nähdä, työpöytäsovelluksen arkkitehtuuri muistuttaa hyvin paljon kuvassa 15 ja 16 esitettyä web-sovelluksen arkkitehtuuria ilman erillistä palvelinta. Kuvassa on myös näytetty päänäkymien kontrollerit, jotka olivat web-sovelluksessa sulautettuina näkymiin Angular-ohjelmistokehyksen periaatteiden vuoksi. Arkkitehtuuri on MVC-mallin mukainen, jossa mallia ja logiikkaa koskevat muutokset ja tapahtumat ilmoitetaan näkymästä kontrollerille, joka sitten ohjaa tapahtuman oikeaan paikkaan ja päivittää näkymää mallista tulevien tietojen mukaan. Näkymät vaihtavat keskenään ainoastaan näkymään ja sen tyyliin liittyvää tietoa, esimerkiksi ikkunan koon muutos ja näkymäkomponentin piilotus. Työpöytäsovelluksessa kolmiulotteisen mallin näyttämiseen käytetään xBIM-liitännäiseen liittyvää xBIM-Presentation kirjastoa.

4.6 BIM-tarkastelutyökalun näkymät

Jotta kokonaisuudesta tulisi eheä, näkymien ja komponenttien vastuualueet ja riippuvuudet tulee olla selkeitä. Tässä aliluvussa on selitetty luvussa 4.4 esitettyjen eri näkymien vastuualueet. Näkymä voi pitää sisällään muita näkymiä.

Juurinäkymän tehtävänä on olla informaation välittäjänä eli mediaattorina objektihierarkian, kolmiulotteisen mallin ja palvelimen välillä. Se välittää tietoa objektihierarkian tapahtumista kolmiulotteiselle mallille ja toisinpäin. Se myös huolehtii tiedoston lataus- ja tallennuspyynnöistä palvelimelle ja vastaanottaa palvelimen käsittelemän BIM-mallin.

Kolmiulotteinen malli -näkymä näyttää palvelimen prosessoiman BIM-mallin kolmiulotteisesti. Kolmiulotteisen mallin näyttämiseen käytetään xBIMin tarjoamia työkaluja, jotka näyttävät mallin niin kuin toiminnallisuuskuvauksessa on määritelty. Kolmiulotteisen mallin on syytä ottaa kiinni halutut xBIMin mallin luomat tapahtumat ja välittää niistä tieto juurinäkymään, joka sitten hoitaa tiedonvälityksen kaikille muille sitä tarvitseville komponenteille. Tällaisia tapahtumia ovat esimerkiksi ohjelman objektin korostamistapahtuma tai uuden mallin lataaminen.

Objektinäkymän vastuulla on näyttää palvelimen prosessoiman mallin objektihierarkia puumaisessa rakenteessa. Sen täytyy myös kommunikoida sekä objektin ominaisuusnäkymälle että juurinäkymälle objektinäkymän muutokset. Objektihierarkia myös hallitsee lajitteluja ja muuntautuu käyttäjän valitsemien suodattimien ja ryhmittelyjen mukaisesti. Sen vastuulla on näyttää käyttäjälle valitut lajittelukriteerit ja antaa yksinkertaiset työkalut muokata niitä.

Objektin ominaisuudet -näkymän vastuulla on olla aina ajan tasalla objektihierarkian valintojen kanssa. Se on myös ajan tasalla kolmiulotteisen mallin valintojen kanssa, sillä malli ilmoittaa juurinäkymän kautta objektihierarkialle käyttäjän ajantasaisimmat valinnat ja objektihierarkia ilmoittaa nämä objektin ominaisuusnäkymälle. Ominaisuusnäkymä tarjoaa myös työkalut objektien lajittelulle, jos käyttäjä haluaa vaihtaa lajittelukriteerejä jonkun tietyn objektin ominaisuuksien perusteella.

Palvelimen julkinen rajapinta on vastuussa kommunikoinnista asiakasohjelman kanssa, autentikaatiosta ja kutsujen ohjaamisesta oikealle palvelulle. Se esimerkiksi ohjaa IFC-tiedoston muuntamispyynnön BIM-palvelulle. BIM-palvelun tehtävänä on keskustella tietokantapääsykerroksen kanssa ja hoitaa BIM-tiedostojen muunnot.

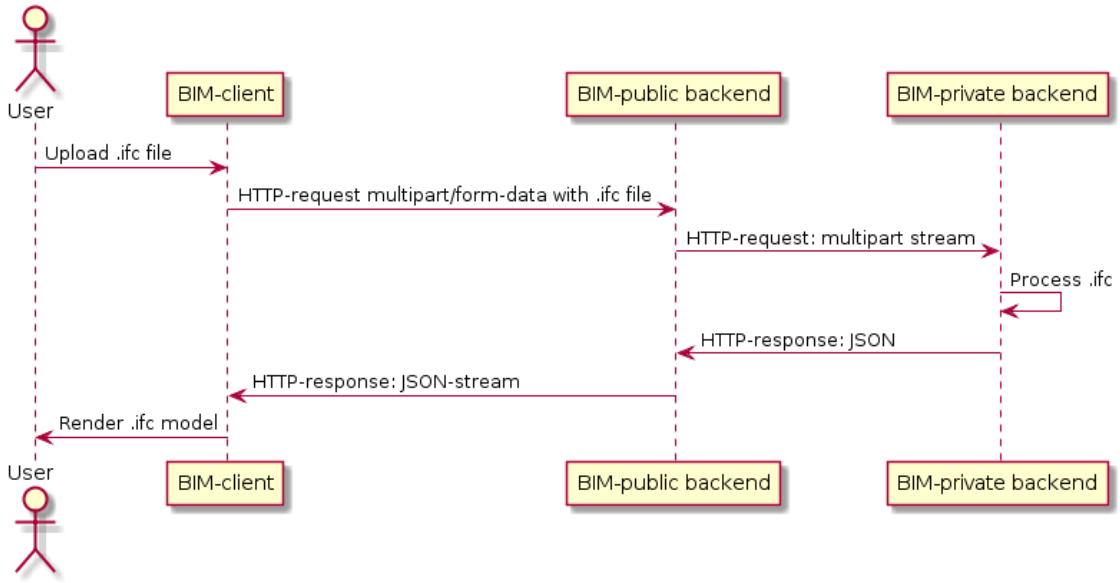
4.7 Open source -komponentin integrointi tarkastelusovellukseen

Suunniteltavassa työpöytäsovelluksessa xBIM-liitännäisen käyttöönotto on hyvin suoraviivaista, sillä xBIM-liitännäinen on toteutettu samalla teknologialla kuin suunniteltava työpöytäsovellus. Liittäminen tulee toimimaan samalla tavalla kuin muutkin .NET Framework -kirjastot: tuomalla liitännäisen kirjastot sovellusten projektiin, sekä käyttämällä niitä samalla tavalla kuin mitä tahansa muuta ohjelmointikirjastoa.

Verkkopohjaisessa sovelluksessa xBIM-liitännäinen otetaan käyttöön kahdessa paikassa: asiakasohjelmassa kolmiulotteisen mallin näyttämiseen ja palvelimella IFC-tiedoston muuntami-

seen. Asiakasohjelmassa liitännäinen toimii helposti, vain liittämällä kirjasto osaksi projektia. Palvelimen puolella sitä ei saa suoraan liitettyä projektiin, sillä palvelin on toteutettu käyttäen .NET Core -ohjelmointikirjastoa, jota liitännäinen ei tue. Ratkaisuna on tehdä erillinen yksityinen palvelin, joka toimii xBIM-liitännäisen tukemalla .NET Framework -ohjelmointikirjastolla ja siirtää tiedosto julkiselta palvelimelta tälle palvelimelle hyväksikäyttäen virtuaalista tiedostovirtaa.

IFC-tiedoston muuntamisessa on otettava huomioon, ettei luotu tiedostovirta varastoi kokonaisia tiedostoja välimuistiin, sillä IFC-tiedostot voivat olla hyvinkin suuria, esimerkiksi jopa 300 megatavua. Tästä tapahtumasta on esitetty sekvenssikaavio kuvassa 17. Tämän operaation suorituskykyä voidaan myös parantaa käyttämällä mahdollisimman paljon asynkronisia metodeja.



Kuva 17. Sekvenssikaavio IFC-tiedoston lataamisesta

Yksityisen palvelun tarkoituksena on saada IFC-tiedostot muunnettua, sellaiseen muotoon, joka voidaan näyttää järkevällä tavalla asiakasohjelman komponenteissa. Se vastaanottaa ja vastaa muuntokäskyihin sekä käyttää xBIM essentials -ja xBIM geometry -kirjastoja apuna tiedostojen muuntamisessa.

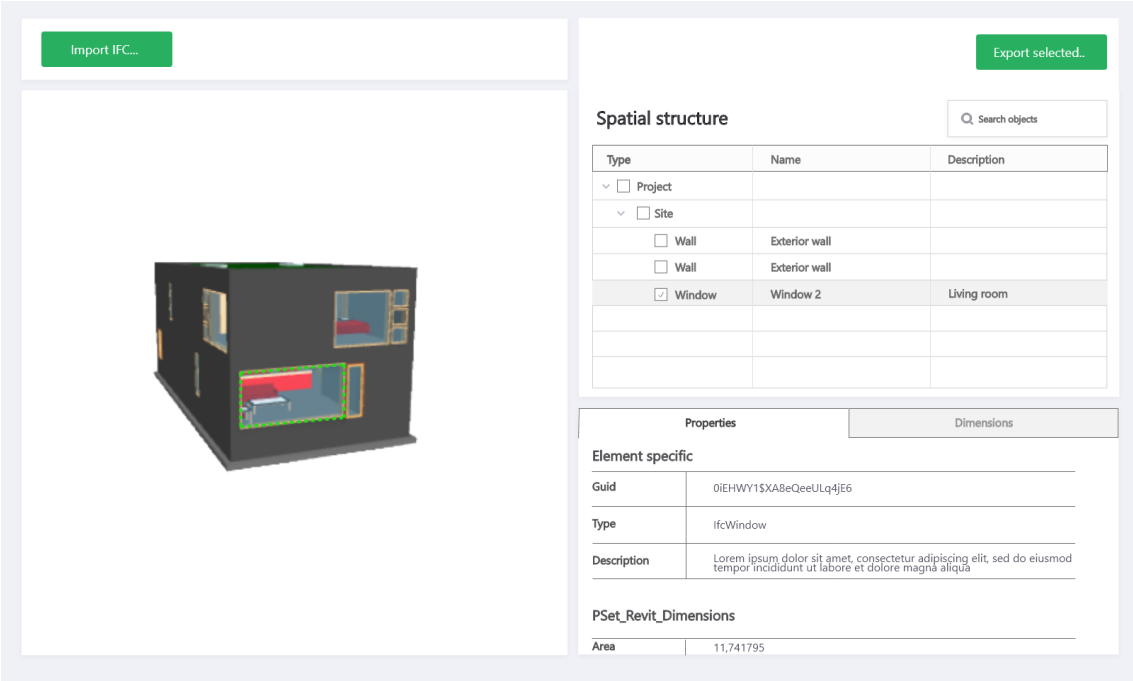
4.8 Tarkastelusovelluksen käyttöliittymä

Käyttöliittymä täytyy suunnitella mahdollisimman yksinkertaiseksi, nopeaksi käyttää ja helposti ymmärrettäväksi käyttäjäkokemuksen parantamiseksi, sillä työkalu tulee olemaan vain aputyökalu, jota ei ole pakko käyttää missään vaiheessa rakennusprosessia. Työkalu tulee siis nopeuttamaan työvirtaa, eikä niinkään tule elintärkeäksi osaksi ohjelmistoperhettä. Käyttöliittymän suunnittelussa tulee ottaa myös huomioon EVRYn toiminnanohjausjärjestelmän sovelluksien ulkoasu ja värimaailma sekä hyvät suunnitteluperiaatteet. Mock-up eli malli verkkopohjaisen BIM-työkalun käyttöliittymän perusnäkymästä on esitetty kuvassa 18. Kuvassa näkyvät kuvan vasemmalla puolella oleva ’kolmiulotteinen malli’-näkymä, oikealla ylhäällä objektihierarkianäkymä ja oikealla alhaalla ’objektin ominaisuus’-näkymä.

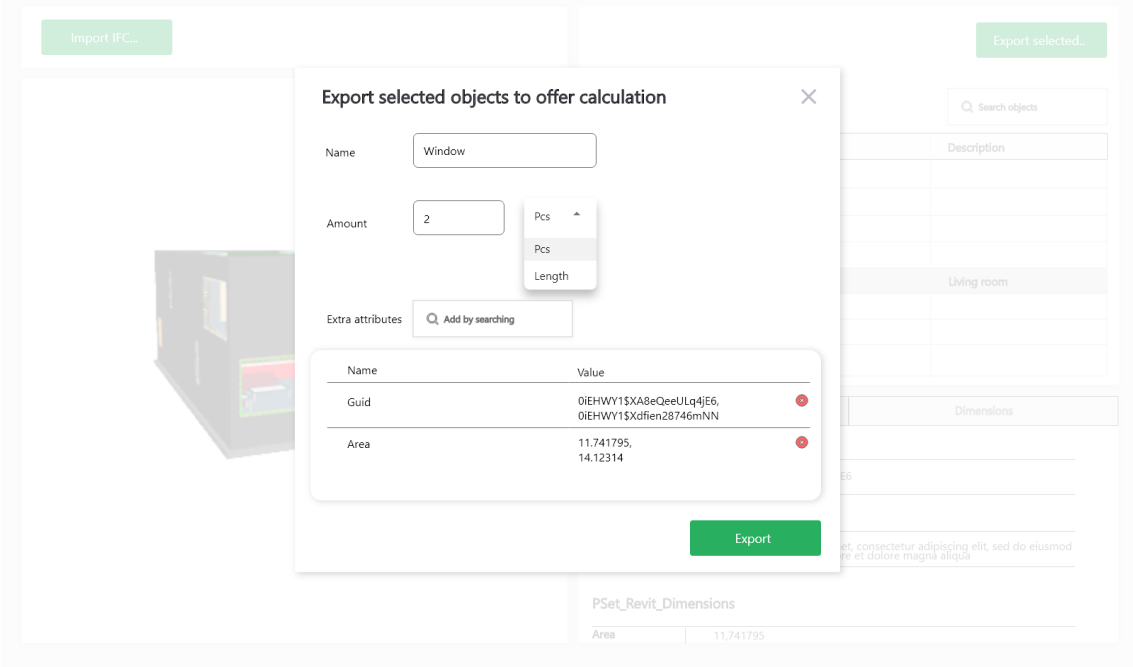
Yksinkertainen BIM-tarkastelutyökalun käyttötapaus voisi olla seuraavanlainen: käyttäjä liikkuu portaalista BIM-työkaluun, joka avaa jo valmiiksi lähetetyn ja muunnetun BIM-mallin perusnäkymään. Tämän jälkeen käyttäjä valitsee hiiren vasemmalla näppäimellä kolmiulotteisesta mallista ikkunan, joka korostuu sekä kolmiulotteisesta mallista että objektihierarkiasta. Samaan aikaan myös objektin tiedot ilmestyvät ominaisuusnäkymään. Käyttäjä valitsee hiiren oikealla näppäimellä ominaisuuden ’Width: 2,41 m’ ominaisuudet-näkymästä ja valitsee ponnahtavasta kontekstivalikosta ’suodata’. Yllä oleva objektihierarkia muuntautuu muotoon, jossa näkyvät vain objektit, joiden leveys on tasan 2,41 metriä. Käyttäjä valitsee kaikki objektit maalaamalla ne hiiren

vasemmalla näppäimellä, jolloin niiden yhteenlasketut ominaisuudet näkyvät ominaisuus-näkymässä. Esimerkiksi ikkunoita ollessa kaksi yhteenlasketuissa ominaisuuksissa näkyisi esimerkiksi ’Count: 2’, ’Width: 4,82’ ja muut yhteiset ominaisuudet yhteenlaskettuina.

Koska tätä työkalua on tarkoitus voida käyttää liitännäisenä, voidaan jo suunnitella miltä objektien ohjelmasta tarjouslaskentaan viemiseen tarkoitettu ikkuna voisi esimerkiksi näyttää. Ikkuna tulisi aueta kuvassa 18 esitetystä ’export selected’-napista. Mock-up -kuva vienti-ikkunasta on esitetty kuvassa 19. Kuvassa 19 ei ole oikeat tietueet tarjouslaskentaa ajatellen vaan niistä on kerrottu tarkemmin aliluvussa 5.2.



Kuva 18. Mock-up käyttöliittymän perusnäkymästä



Kuva 19. Mock-up objektien viennistä

5. BIM-TYÖKALUN LIITTÄMINEN TARJOUSLASKENTASOVELLUKSEEN

Jotta BIM-työkalu voi olla yhteydessä toiseen sovellukseen, on niiden välille luotava jonkinlainen yhteys. Tässä työssä BIM-työkalu liitetään osaksi jo olemassa olevaa Windowsin työpöydällä toimivaa tarjouslaskentasovellusta. Yhteys on syytä suunnitella siten, että jo liitettävään sovellukseen tarvitsee tehdä mahdollisimman vähän muutoksia, jotta se toimisi BIM-työkalun kanssa yhteistyössä. Prioriteettina on saada tietoa siirrettyä yhdensuuntaisesti BIM-työkalulta tarjouslaskentatyökalulle, mutta kahdensuuntainen tiedonsiirto on lopullinen tavoite, jos se vain on mahdollista.

Kuten edellisen luvun alussa mainittiin, tässä työssä päädyttiin lopulta suunnittelemaan ja toteuttamaan kaksi erillistä BIM-työkalua: toinen toimimaan web-ympäristössä ja toinen toimimaan Windowsin työpöydällä. Pääasiallisena syynä tähän jakoon oli se, että web-portaali, johon alkuperäinen BIM-työkalu suunniteltiin, on eri tuotetta kuin tarjouslaskentasovellus, johon tässä työssä on tarkoitus luoda yhteys BIM-työkalusta. Molemmat työkalut on suunniteltu lähes identtisiksi, samoja arkkitehtuureja ja liitännäisiä käyttäen. Erona sovellusten suunnitelmien välillä on työpöytäsovelluksesta erillisen palvelimen puuttuminen. Tässä luvussa käydään läpi arkkitehtuurivaihtoehtoja, joilla alkuperäisen suunnitelman mukainen web-työkalusta yhteys työpöytäsovellukseen olisi ollut mahdollista, sekä lopullinen arkkitehtuurisuunnitelma.

5.1 Integraation vaatimukset

BIM-työkalun ja tarjouslaskentasovelluksen integraation käyttötapausvaatimukset on esitetty seuraavassa luettelossa:

1. Tarjouslaskija voi avata BIM-työkalun tarjouslaskentasovelluksesta.
2. Tarjouslaskija voi siirtää BIM-työkalussa valittujen objektien määrä -ja muun hyödynnettävissä olevan tiedon tarjouslaskentaan tuoterakenteena tai suoritteena.
3. Tarjouslaskija voi valita mitä attribuutteja objekteista käytetään objektien määrinä ja ominaisuuksina.
4. Tarjouslaskija voi päivittää tietyn tuoterakenteen tai suoritteen määrä -tai muita tietoja BIM-työkalusta.
5. Tarjouslaskija voi valita yhdistetylle objektille tunnisteen alun tarjouslaskentasovelluksessa määritetystä tunnistenimikkeistöstä.
6. Tarjouslaskija voi tallentaa muutokset tietokantaan painamalla ’tallenna’-nappia tarjouslaskennassa.
7. Jos tietty objekti on jo viety tarjouslaskentaan, tarjouslaskijan korostaessa objektia, siihen liitetty tuoterakenne/suorite korostuu tarjouslaskennassa.
8. Jos tietty tuoterakenne/suorite on tuotu tai päivitetty viemällä BIM-työkalun objekteja, tarjouslaskijan korostaessa tuoterakenteen/suoritteen siihen liitetyt objektit korostuvat myös BIM-työkalussa.

Näistä käyttötapausvaatimuksista voidaan muodostaa käyttäjätarinat aliluvun 4.1 tapaan, jotka on esitetty seuraavassa luettelossa:

1. Avata BIM-työkalu tarjouslaskentasovelluksesta.
2. Siirtää objektien ominaisuuksia tarjouslaskentaan uudeksi tuoterakenteeksi.
3. Siirtää objektien ominaisuuksia tarjouslaskentaan uudeksi suoritteeksi.
4. Valita määränä käytettävän ominaisuuden vietäessä objekteja.
5. Päivittää tarjouslaskennan tuoterakenteen tietoja objektien ominaisuuksilla.
6. Päivittää tarjouslaskennan suoritteen tietoja objektien ominaisuuksilla.
7. Valita tunnisteen alun vietävän objektin ominaisuuksille jo määritetystä tunnistenimikkeistöstä.

8. Viedä tarjouslaskentaan tulleet muutokset tietokantaan, kun ’tallenna’-nappia painetaan.
9. Korostaa tuoterakenne tarjouslaskennassa valitsemalla objekti.
10. Korostaa suorite tarjouslaskennassa valitsemalla objekti.
11. Korostaa objekti valitsemalla tarjouslaskennassa tuoterakenne.
12. Korostaa objekti valitsemalla tarjouslaskennassa suorite.

Näiden käyttäjätarinoiden perusteella voidaan alkaa suunnitella tarkempaa toiminnallisuutta sekä arkkitehtuuria ratkaisulle. Integraation tarkoituksena on helpottaa ja nopeuttaa tarjouslaskentaohjelman käyttöä, jos rakennusprojektista on luotu jo valmiiksi BIM-malli. Jo aiemmin syötettyjä tietoja ei tarvitse syöttää manuaalisesti uudelleen vaan ne pystyy helposti valitsemaan ja siirtämään BIM-mallista tarjouslaskentaohjelmaan.

5.2 Toiminnallisuus

Käyttäjätarinoiden perusteella tarkempi toiminnallisuus voidaan jaotella neljäksi isoksi kokonaisuudeksi: BIM-työkalun ja mallin avaaminen tarjouslaskentasovelluksesta, objektien vieminen BIM-työkalusta tarjouslaskentasovelluksen riveiksi, objektien korostaminen BIM-työkalussa valitsemalla rivi tarjouslaskennassa ja rivien korostaminen tarjouslaskennassa valitsemalla objekti BIM-työkalussa.

BIM-työkalu voidaan avata tarjouslaskennassa painamalla kuvassa 19 nähtävää ylävalikon ’työkalut’-nappia ja tämän jälkeen valitsemalla vaihtoehto ’Avaa BIM-työkalu’. Jos käyttäjä ei ole ladannut yhtään BIM-mallia työkaluun, näytetään käyttäjälle valikko, jossa kysytään, haluaako käyttäjä ladata työkaluun mallin.

Aliluvussa 4.6 esitettyyn tapaan objektien paketoimisen jälkeen käyttäjä voi aloittaa objektien viennin painamalla ’vie’-nappia. Kuvassa 18 näytettyjen ominaisuuksien lisäksi yhdistetyn objektin tyypiksi pitää pystyä valitsemaan onko kyseessä tuoterakenne vai suorite. Valittaessa tuoterakenne objektille pitää pystyä määrittämään tarjouslaskennan koodi, selite ja määrä. Jos tarjouslaskennassa on valmiiksi korostettuna rivi, laitetaan sen koodi valmiiksi koodikenttään. Tuoterakennetta vietäessä tarkistetaan, onko samalla koodilla olemassa tarjouslaskennassa jo tuoterakenne. Tuoterakenteen ollessa jo olemassa lisätään viedyt määrätiedot sille, muuten tehdään uusi tuoterakenne-rivi BIM-työkalussa valittujen tietojen mukaan.

Vientivalikosta valittaessa suorite objektille pitää pystyä määrittelemään tarjouslaskennan tuoterakenteen koodi, suoritteen koodi, selite ja määrä. Kuten tuoterakenteellakin tarjouslaskennasta jonkin tuoterakenteen tai suoritteen ollessa korostettuna täytetään niitä vastaavat kentät valmiiksi ja päivitetään olemassa olevaa suoritetta sen ollessa jo olemassa. Suoritteen koodin alun voi myös valita tarjouslaskentasovelluksessa määritellystä suoritteiden nimikkeistöstä, joka kertoo suoritteen tyypin. Jos suoritteen koodin alku valitaan nimikkeistöstä, sen loppuosa generoidaan olemaan seuraava vapaa koodi olemassa olevien suoritteiden perusteella.

Tuoterakenteen tai suoritteen tapauksessa määrän voi laskea valmiiksi halutun objektien yhteisen ominaisuuden, esimerkiksi pituuden tai määrän, perusteella. Kun haluttuja tietoja viedään tarjouslaskentaan niitä ei saa tallentaa tietokantaan valmiiksi vaan ne täytyy näkyä tarjouslaskennassa vain paikallisina tallentamattomina riveinä.

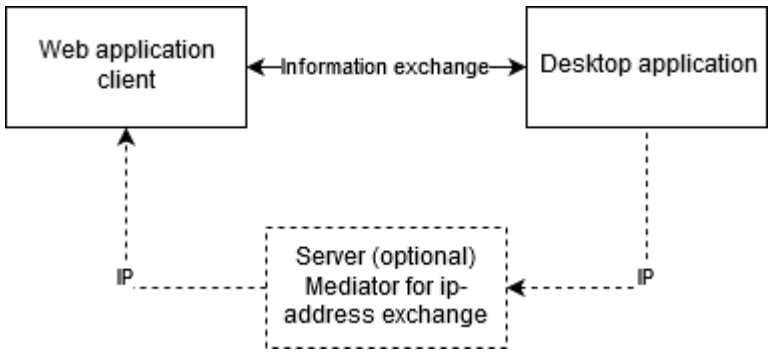
Kun BIM-työkalussa korostetaan jokin objekti tai joitakin objekteja, tehdään tarkistus, jossa tarkistetaan, onko objekteja jo viety tarjouslaskentaan. Jos objektit on viety tarjouslaskentaan, lähetetään BIM-työkalusta pyyntö korostaa valittuja objekteja vastaavat tuoterakenteet tai suoritteet tarjouslaskennassa. Tarjouslaskennasta rivejä korostaessa lähetetään BIM-työkalulle korostuspyyntö valituista tuoterakenteista tai suoritteista. Jos korostuspyynnössä olevat tuoterakenteet tai suoritteet on viety BIM-työkaluun tarjouslaskennasta, korostetaan niitä vastaavat objektit BIM-työkalussa. Jos BIM-työkalua ei ole avattu tarjouslaskennasta, tarjouslaskennassa riviä korostaessa ei tulisi tapahtua mitään toiminnallista muutosta jo olemassa olevaan toimintaan verrattuna.

5.3 Web-sovelluksen integrointiarkkitehtuurivaihtoehtojen vertailu

Vaikka web-pohjaista BIM-sovellusta ei loppujen lopuksi integroidakaan tarjouslaskentaohjelmaan, on silti tärkeää käydä läpi vaihtoehdot, kuinka se olisi ollut mahdollista tulevaisuutta varten. Tässä aliluvussa käydään läpi vaihtoehtoja miten web-pohjainen sovellus voisi kommunikoida työpöytäsovelluksen kanssa ja miten BIM-työkalun ja tarjouslaskentasovelluksen integraatio olisi ollut mahdollista. Nämä vaihtoehdot toimivat referenssinä tulevaisuutta varten, jos web-portaali ja tarjouslaskentasovellus yhdistetään yhtenäiseksi tuotteeksi.

5.3.1 Peer-to-peer

Ensimmäinen vaihtoehto on välittää informaatiota suoraan kahden ohjelman välillä eli niin sanottu *peer-to-peer* -tiedonvälitystapa, jonka arkkitehtuuri on esitetty kuvassa 20. Tässä vaihtoehdossa toinen sovelluksista nostetaan palvelimeksi, johon sitten voidaan ottaa yhteys toisesta ohjelmasta. Tämän vaihtoehdon hyvänä puolena on se, että erillistä palvelinta ei tarvita, vaan tietoa saadaan siirrettyä täysin kahden ohjelman välillä. Huonona puolena on, että ohjelmien täytyy tietää toistensa IP-osoitteet (*engl. Internet Protocol*) eli osoitteet, joissa laitteet sijaitsevat verkossa. Tämän tiedon voi välittää joko manuaalisesti käsin syöttämällä ohjelmiin osoitteet tai käyttämällä erillistä palvelinta, josta toinen ohjelma saa toisen ohjelman osoitteen. [35] s.9-12

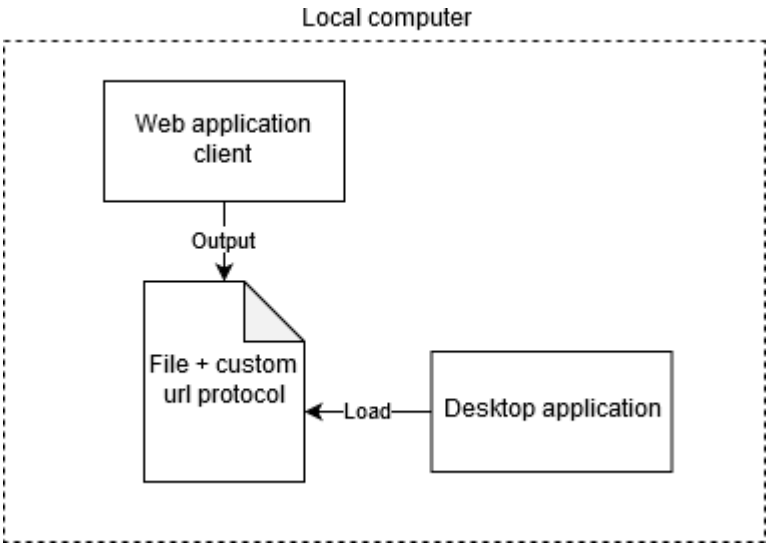


Kuva 20. Peer-to-peer yhteys

5.3.2 Paikallinen tiedonsiirto

Toinen arkkitehtuurivaihtoehto on hyvin yksinkertainen: paikallinen tiedonsiirto. Paikallisessa tiedonsiirrossa kaikki informaatio liikkuu paikallisen laitteen, jossa tarjouslaskentaohjelma sijaitsee, lävitse. Tietoa välitetään tiedostoina ja esimerkiksi käyttämällä verkkoselaimen *custom url* -protokollaa, jolla pyydetään tietokonetta avaamaan tiedosto tietyllä ohjelmalla. Arkkitehtuurikuva tästä tavasta on esitetty kuvassa 21.

Vaihtoehtona tähän olisi upottaa tarjouslaskentasovellukseen selain ja kommunikoida tarjouslaskentasovellukseen upotetun verkkoselaimen lävitse erilaisilla suoritettavilla komentosarjoilla. Hyvänä puolena tässä tavassa olisi riippumattomuus verkon toiminnasta verkkosovelluksen lataamisen jälkeen. Huonona puolena tässä vaihtoehdossa olisi kahdensuuntaisen tiedonsiirron vaikeus, kankeus ja huono käyttäjäystävällisyys. Huono käyttäjäystävällisyys johtuu upotetusta selaimesta tai siitä, että käyttäjä pakotetaan pitämään molemmat sovellukset yhtäaikaisesti auki. [26]



Kuva 21. Paikallisen tiedonsiirron arkkitehtuuri

5.3.3 Viestinvälittäjä

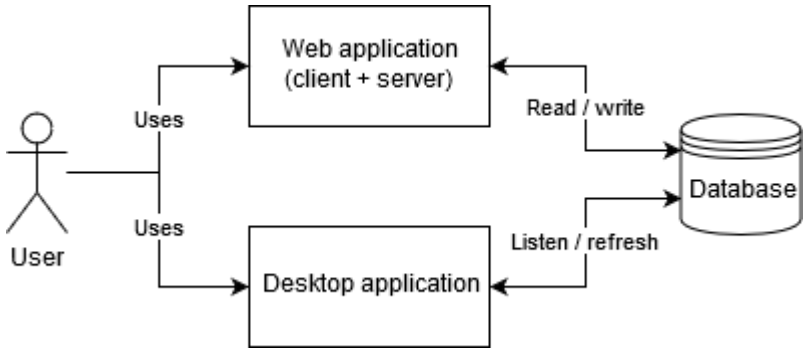
Kolmantena vaihtoehtona on käyttää niin sanottua message broker eli viestinvälittäjä -arkkitehtuuria. Viestinvälittäjä arkkitehtuurissa on julkaisijoita ja tilaajia, jotka voivat olla suoraan yhteydessä toisiinsa, mutta yleensä niiden välistä tiedonsiirtoa hallinnoi palvelin. Ideana on, että tilaajat liittyvät kuuntelemaan jonkin tyyppistä tapahtumaa, jota julkaisijat sitten julkaisevat. Julkaisijan julkaistessa tietoa tapahtumana tieto välitetään jokaiselle tilaajalle, joka kuuntelee kyseistä tapahtumaa. Tämä tiedonvälitysvaihtoehto on havainnollistettu kuvassa 22. [33]

Hyvänä puolena on tässä, että palvelimen ei tarvitse tallentaa tietoa tietokantaan ja se voi poistaa tiedon sen jälkeen, kun jokainen tilaaja on sen saanut eli niin sanotusti kuluttanut. Ratkaisu on myös hyvin skaalautuva ja pystyy toimimaan asynkronisesti eli vastaamaan yhtäaikaisesti useaan eri pyyntöön eri säikeillä myös hyvin suurella pyyntömäärällä. Huonona puolena tässä on, että vanhan web-portaalissa käytetyn arkkitehtuurin päälle joutuisi luomaan täysin uuden arkkitehtuurin, joka käyttää viestinvälitys-arkkitehtuuria. Esimerkki avoimen lähdekoodin viestinvälittäjästä on RabbitMQ [33], jota esimerkiksi Instagram käyttää palvelussaan [34].

Kuva 22. Message broker-arkkitehtuuri

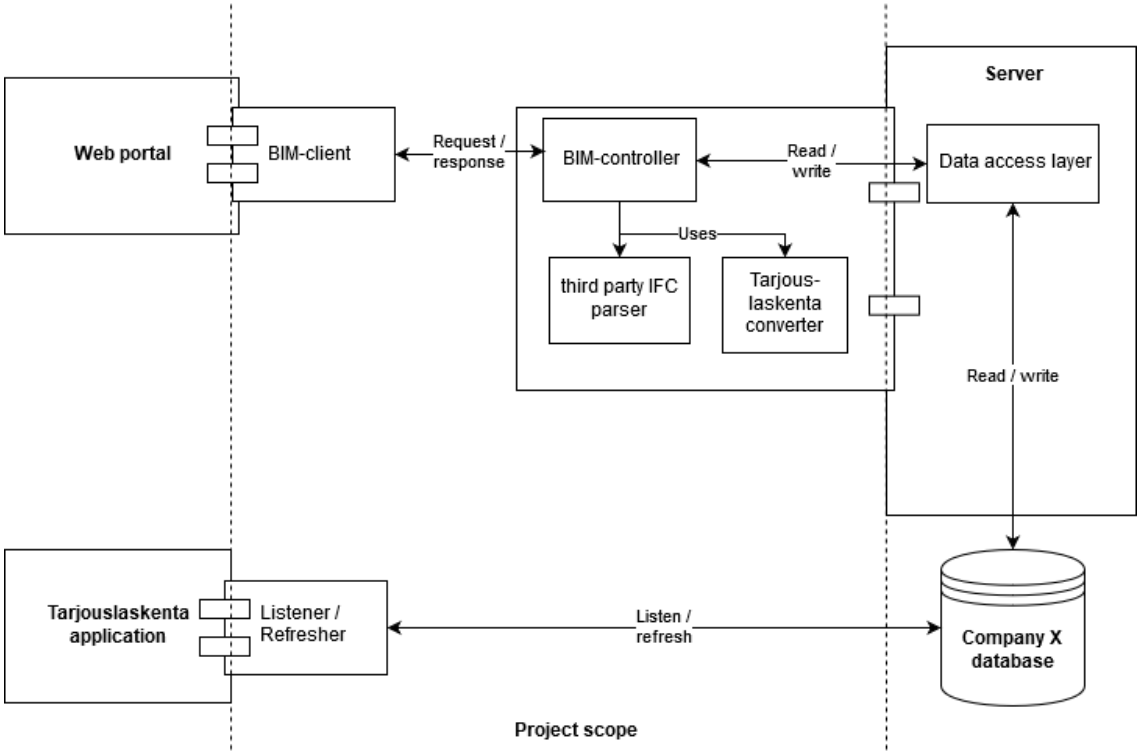
5.3.4 Asiakasohjelma-palvelin-tietokanta-työpöytäsovellus

Neljäs arkkitehtuuri olisi luonnollinen vaihtoehto nykyisen arkkitehtuurin huomioon ottaen, koska web-sovellus ja työpöytäsovelluksen ainut yhteinen yhteys on käytössä oleva tietokanta. Tieto kulkisi web-sovelluksen asiakasohjelmasta palvelimen kautta tietokantaan, jonka muutoksia voisi sitten kuunnella työpöydällä olevasta tarjouslaskentasovelluksesta. Yleiskuva tästä arkkitehtuurista on esitetty kuvassa 23.



Kuva 23. Yleiskuva asiakasohjelma-palvelin-tietokanta-työpöytäsovellusarkkitehtuurista

Tässä vaihtoehdossa tietokantaa käytettäisiin tiedonvälittäjänä, jonka vuoksi ongelmaksi voisi muodostua vaatimus, että tietoa ei saa tallentaa tietokantaan ennen kuin käyttäjä painaa 'Tallenna'-nappia tarjouslaskentasovelluksessa. Tämän ongelman voisi kuitenkin ratkaista palaamalla tietokannan tilaan ennen BIM-työkalusta tietojen vientiä tarjouslaskennan osalta tietojen haun jälkeen tai tallentamalla BIM-työkalusta tulleen tiedon erilliseen tauluun tietokannassa ennen tallentamista. Hyvänä puolena tässä ratkaisussa olisi se, että nykyiseen arkkitehtuuriin tarvitsisi todella vähän muutoksia tämän toimimiseksi. Huonona puolena olisi se, että tiedonvälitys olisi reaaliaikaista vain yhdensuuntaisesti tai tilaa pitäisi molemmissa ohjelmissa päivittää hyvin usein. Tämän ollessa hyvin realistinen vaihtoehto pienillä resursseilla toteutettavaksi tämä arkkitehtuuri on esitetty tarkemmin kuvassa 24.

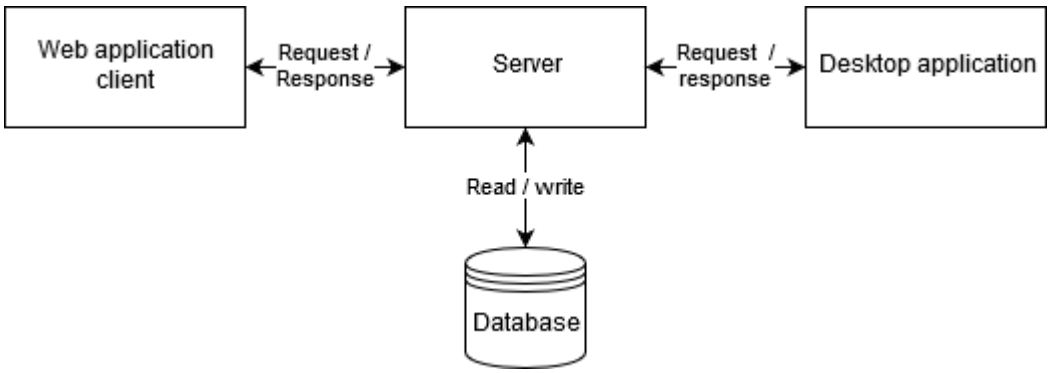


Kuva 24. Tarkempi kuvaus asiakasohjelma-palvelin-tietokanta-työpöytäsovellusarkkitehtuurista

5.3.5 Asiakasohjelma-palvelin

Viidentenä ja viimeisenä tässä työssä tutkittuna vaihtoehtona on perinteinen asiakasohjelma-palvelin -arkkitehtuuri, joka on esitetty kuvassa 25. Jo olemassa oleva web-portaali, johon web-pohjainen BIM-työkalu liitetään, on toteutettu jo käyttämällä tätä arkkitehtuurimallia. Tämä arkkitehtuuri on jatkumo aliluvussa 5.4.4 esitetylle arkkitehtuurille ja saataisiin toteutettua muuttamalla työpöytäsovelluksen arkkitehtuuri käyttämään palvelinta tiedon hakemiselle tietokannan sijaan. Tämän hyötynä olisi reaaliaikainen kaksisuuntainen tiedonsiirto palvelimen ja työpöytäsovelluk-

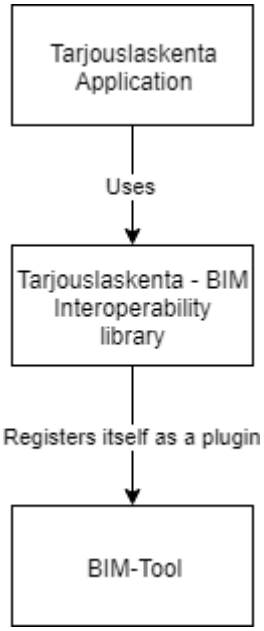
sen välillä esimerkiksi SignalR-liitännäistä hyväksikäyttämällä [25]. Huonona puolena on, että tarjouslaskenta-työpöytäsovelluksen arkkitehtuuria pitäisi muuttaa radikaalisti tai luoda toinen arkkitehtuuri vanhan päälle. [43]



Kuva 25. Asiakasohjelma-palvelin -arkkitehtuuri

5.4 Ratkaisun arkkitehtuuri

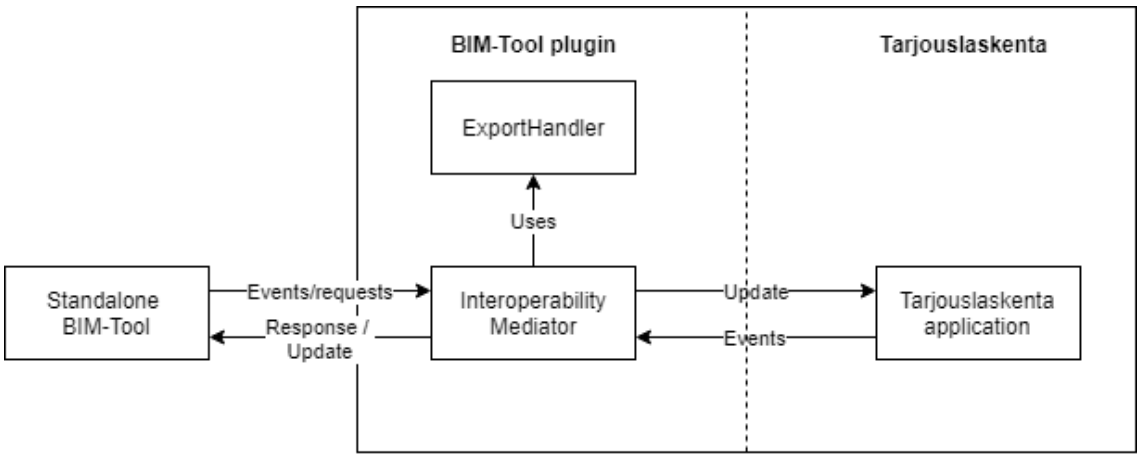
Lopullinen tarjouslaskentasovellukseen integroitava ratkaisu päädyttiin siis toteuttamaan työpöytäsovelluksena. Tämän lisäksi toteutetaan tulevaisuutta ja BIM-mallin tarkastelua varten websovellus, jota ei ole tarkoitus integroida osaksi tarjouslaskenta-työpöytäsovellusta. Tavoitteena oli tuottaa mahdollisimman yksinkertainen ratkaisu, joka vaatii mahdollisimman vähän muutoksia jo olemassa olevaan tarjouslaskentaohjelmaan. Ratkaisuksi tähän voidaan toteuttaa erillinen BIM-työkaluyhteistyömoduuli, jota tarjouslaskentaohjelma käyttää BIM-työkalun kanssa kommunikoimiseen ja toisinpäin. Yhteistyömoduuli toimii siis mediaattorina näiden kahden sovelluksen välillä. Käytännössä siis tarjouslaskentasovellus avaa BIM-työkalun napista käyttämällä yhteistyökirjastoa ja rekisteröi yhteistyömoduulin komponentiksi, johon BIM-työkalussa tapahtuvat tapahtumat ilmoitetaan. Yhteistyömoduulin avattua BIM-työkalun, se rekisteröi itsensä BIM-työkalun liitännäiseksi, jonka jälkeen BIM-työkalu tietää ilmoittaa tarjouslaskentaa koskevista tapahtumista tälle yhteistyömoduulille. Tämä on havainnollistettu kuvassa 26.



Kuva 26. Yhteistyökirjaston käyttö

Rekisteröimisen jälkeen molemmat sovellukset voivat käyttää yhteistyömoduulia mediaattorina tiedonvälitykseen, mikä on havainnollistettu kuvassa 27. BIM-työkalu käyttää yhteistyömoduulia sen tapahtumien, esimerkiksi objektivalinnan muuttumisen, ilmoittamiseen ja haluttujen tie-

tojen, esimerkiksi jo vietyjen objektien tunnisteiden, hakemiseen. Tapahtumailmoituksen saapuessa yhteistyömoduulille se päivittää tarjouslaskentasovelluksen tietoja tapahtumien kuvaamalla tavalla. Tarjouslaskentasovellus voi myös käyttää yhteistyömoduulia sen tapahtumien, esimerkiksi rivin valinnan muuttumisen, ilmoittamiseen. BIM-työkalu pyytää myös yhteistyömoduulilta vientikäsittelijän, joka hoitaa objektien tarjouslaskentaan viemisen käyttöliittymäikkunan ja itse viemisen. BIM-työkalu toimittaa vientikäsittelijälle sen haluamat tiedot vietävistä objekteista.



Kuva 27. Yhteistyökomponentti mediaattorina

Tällainen arkkitehtuuri mahdollistaa sen, että BIM-työkalun voi liittää mihin tahansa sovellukseen, joka on toteutettu samalla teknologialla. Liittäessä BIM-työkalua on vain huolehdittava siitä, että yhteistyökirjaston täytyy toteuttaa BIM-työkalun käyttämä rajapinta. Tämä arkkitehtuuri on hyvin paljon asiakasohjelma-palvelinarkkitehtuurin kaltainen, jos yhteistyökirjastoa ajatellaan palvelimen liitännäisenä, joka toimittaa tietoa erinäisillä kutsuilla ja sen nostamia tapahtumia voi kuunnella.

5.5 Komponentit

Tehtävä työ voidaan jakaa kahteen kokonaisuuteen: uudet luotavat komponentit ja muutokset jo olemassa oleviin komponentteihin. Uusina komponentteina luodaan yhteistyökirjasto ja sen käyttämä vientikäsittelijä. BIM-työkaluun sekä tarjouslaskentasovellukseen joutuu tekemään pieniä muutoksia yhteistyökirjastoon käyttöönottamiseksi. Tässä aliluvussa on esitelty uudet komponentit sekä jo olemassa olevien komponenttien muutokset.

5.5.1 Yhteistyömoduuli

Yhteistyömoduulin tehtävänä on nimensä mukaisesti mahdollistaa tarjouslaskentasovelluksen ja BIM-työkalun yhteistyö. Sen vastuulla on tarkistaa tapahtumissa ja pyynnöissä, että molemmat osapuolet ovat läsnä ja valmiita aloittamaan keskustelun keskenään. Sen tehtävänä on myös muistaa BIM-työkaluista jo pois vietyjen objektien tunnisteet objektien korostamisen mahdollistamiseksi ja vietyjen objektien pois suodattamisen näkymästä mahdollistamiseksi.

Yhteistyömoduulin pitää mahdollistaa seuraavassa luettelossa esitetyt tapahtumat BIM-työkalulta tarjouslaskentasovellukseen:

1. Avaa vientikäsittelijän vienti-ikkuna.
2. Vie objektien tiedot tietyllä tunnisteella olevaksi tuoterakenteeksi. Jos tuoterakenne on olemassa, päivitetään määrätietoja. Jos tuoterakenne ei ole olemassa, luodaan uusi tuoterakenne vientikäsittelijästä saatavien tietojen mukaan.
3. Vie objektien tiedot tietyllä tunnisteella olevaksi suoritteeksi. Jos suorite on olemassa, päivitetään määrätietoja. Jos suorite ei ole olemassa, luodaan uusi suorite vientikäsittelijästä saatavien tietojen mukaan.
4. Korosta näitä objekteja vastaava tuoterakenne tai suorite, jos se on olemassa.

Näiden lisäksi yhteistyömoduulin tulee mahdollistaa BIM-työkalusta objektin korostaminen, joka vastaa valittua tuoterakennetta tai suoritetta, jos sellainen on olemassa. Yhteistyömoduulista pitää myös pystyä avaamaan BIM-työkalu ja luomaan siihen yhteys. Yhteistyömoduulin olisi myös hyvä mahdollistaa xBIM-liitännäisellä jäsenneltyjen tiedostojen sekä vietyjen objektien tunnisteiden tallennus käyttäen tarjouslaskennan tietokantamoduulia.

5.5.2 Vientikäsittelijä

Vientikäsittelijän tehtävänä on antaa käyttäjälle visuaaliset työkalut valittujen objektien vientiä varten. Se pitää sisällään vienti-ikkunan, joka visualisoi käyttäjälle kaikki viemistä koskevat tiedot. BIM-työkalu voi pyytää yhteistyömoduulia avaamaan tämän vienti-ikkunan, jolloin valittujen objektien vieminen voi alkaa. Vienti-ikkunan tulee toimia aliluvussa 5.2 kuvatulla tavalla ja sen käyttöliittymä on kuvattu aliluvussa 5.6.

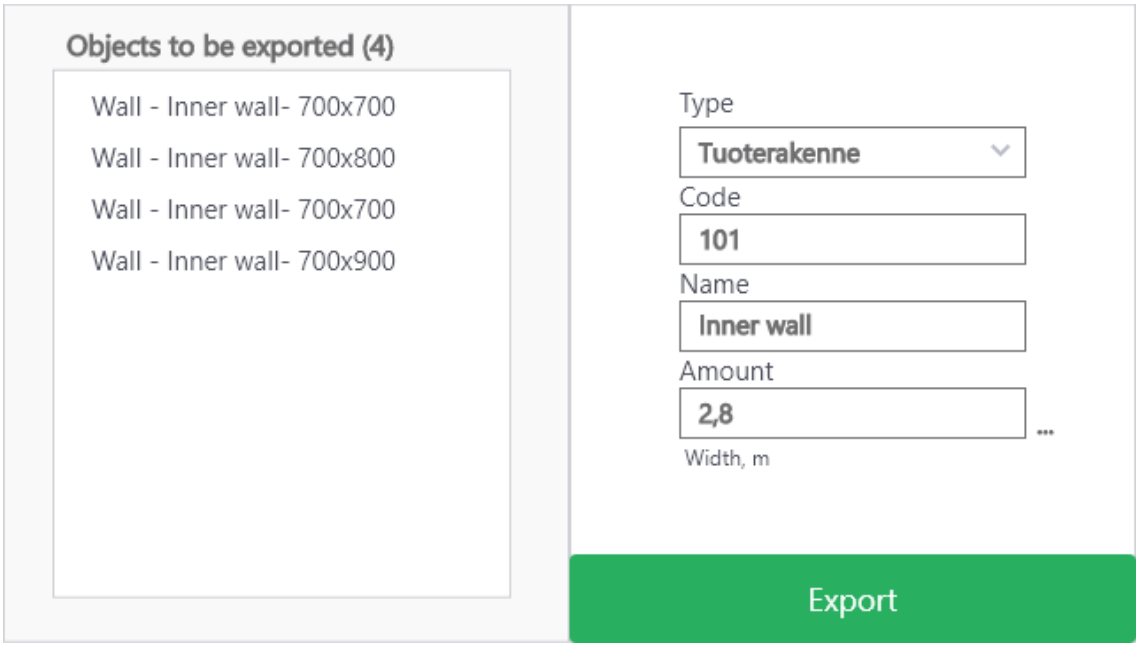
5.5.3 Muutokset sovelluksiin

BIM-työkalun ollessa valmis integrointia varten hyvin pienet muutokset riittävät yhteistyön mahdollistamiseksi. Tämä tulisi onnistua laittamalla pyyntö yhteistyömoduulille moduulia kiinnostavista tapahtumista, eli vienti-ikkunan avaus, objektien vienti ja objektin korostus.

Tarjouslaskentasovellus on erittäin suuri kokonaisuus jo tuotannossa olevaa koodia, jossa ei ole selkeitä rajapintoja kaikille tarvittaville ominaisuuksille. Tämän takia yhteistyömoduuli suunniteltiin tarjouslaskentasovelluksesta erilliseksi moduuliksi, ennemmin kuin aloitettiin muokkaamaan ja luomaan rajapintoja jo olemassa olevan sovelluksen sekaan uusien riippuvuuksien minimoimiseksi. Ainoastaan BIM-työkalun aukaiseminen joudutaan ilmoittamaan suoraan yhteistyömoduulille. Muut toiminnot onnistuvat kuuntelemalla tarjouslaskentatyökalun tapahtumia yhteistyömoduulissa.

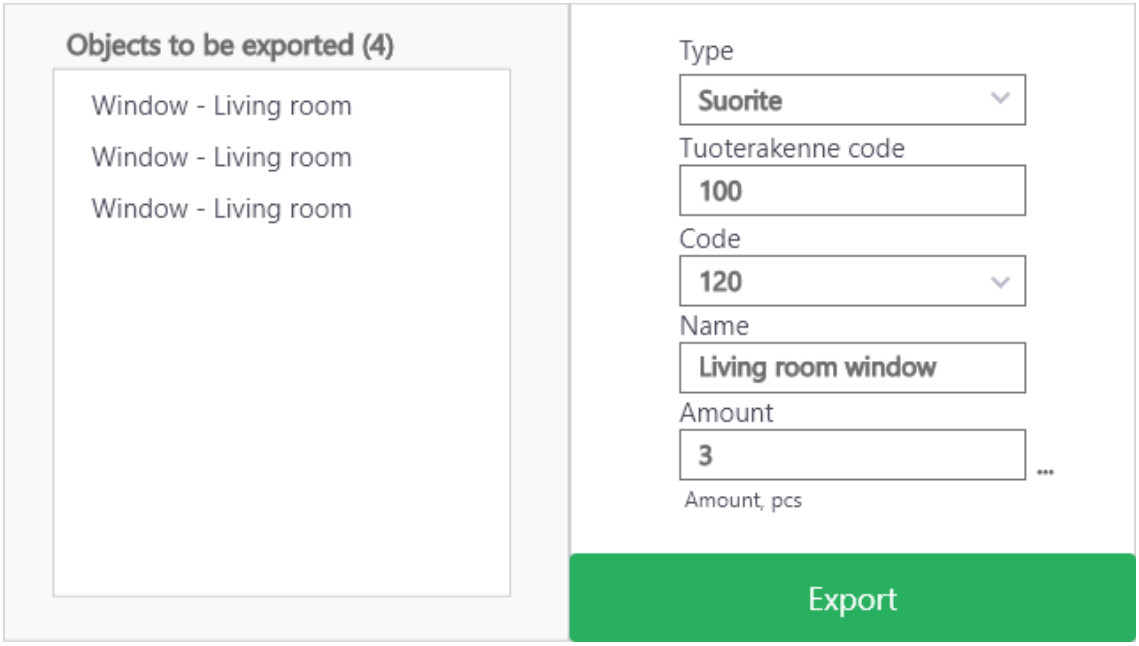
5.6 Vientikäsittelijän käyttöliittymä

Vientikäsittelijän tulee näyttää käyttäjälle yksinkertaiset tiedot kaikista objekteista, jotta käyttäjä voi tarkistaa nopeasti onko hän viemässä oikeita objekteja. Tämän lisäksi käyttäjän täytyy pystyä valitsemaan luvussa 5.2 esitetyt tiedot vietävälle objektille tai objektiryhmälle. Mock-up -kuva vientikäsittelijän vienti-ikkunasta tuoterakenteen tapauksessa on esitetty kuvassa 28 ja suoritteen tapauksessa kuvassa 29.



Kuva 28. Vientikäsittelijän ikkuna – tuoterakenne

Vientikäsittelijän tulee myös osata hakea tarvitsemansa tiedot, esimerkiksi tarjouslaskentasovelluksessa korostetun tuoterakenteen, yhteistyömoduulilta. Nämä tarvittavat tiedot oli esitelty luvussa 5.2



Kuva 29. Vientikäsittelijän ikkuna – suorite

Yksinkertainen käyttötapaus vientikäsittelijästä tuoterakenteen tapauksessa voisi olla: käyttäjä maalaa kuvan 28 vasemmalla puolella esitetyt seinät BIM-työkalusta, jonka jälkeen hän valitsee ’vie’. Tämän jälkeen aukeaa kuvan 28 mukainen ikkuna, jossa on kentissä ’code’ ja ’name’ valmiiksi täytettynä tarjouslaskennassa korostetun tuoterakenteen tiedot. Määrä on arvattu tarjouslaskennassa korostetun rivin yksikön mukaan esimerkiksi ensimmäiseksi ominaisuudeksi, jonka yksikkö on metri. Käyttäjä haluaa luoda uuden tuoterakenteen seinistä, hän pyyhkii ’code’-kentässä olevan arvon, jolloin kenttä ’name’ palautuu parhaaksi arvaukseksi, joka on arvattu objektien nimien yhteisestä osasta. Kenttä ’määrä’ palautuu myös oletusarvoon eli kappaleisiin, joita tässä tapauksessa on neljä. Käyttäjä syöttää uuden koodin ’101’ ’code’-kenttään, jonka jälkeen hän painaa ’määrä’-kentän vieressä olevaa ’…’-nappia. Nappi avaa kontekstivalikon, jossa näkyvät kaikki objektien yhteiset määrälliset ominaisuudet. Käyttäjä valitsee kontekstivalikosta ’width, m’, jolloin ’amount’-kenttään generoituu objektien yhteenlaskettu määrä. Lopuksi käyttäjä klikkaa ’vie’-nappia, jolloin tarjouslaskentaan ilmestyy rivi, jossa koodi on 101, nimi on ’Inner wall’, määrä on 2,8 ja yksikkö on ’m’.

Toinen yksinkertainen käyttötapaus vientikäsittelijästä voisi olla kuvan 29 mukainen tilanne, jossa käyttäjä on valinnut BIM-työkalusta kuvatut ikkunat, avannut vientivalikon ja vaihtanut tyypin suoritteeksi. Kentät täyttyvät samanlailla valmiiksi kuin tuoterakenteenkin tapauksessa. Käyttäjä painaa nuolta ’code’-kentän vieressä, jolloin ohjelma hakee tarjouslaskennasta suoritteiden nimikkeistön eli koodien alkupätkät, jotka kuvaavat suoritteen tyyppiä. Käyttäjä valitsee alasvetovalikosta kentän, jossa lukee ’12 – windows’, jolloin kenttään generoituu ensimmäinen vapaa koodi alulla ’12’. Käyttäjä klikkaa ’vie’-nappia, jolloin tarjouslaskentaan ilmestyy uusi suorite tuoterakenteen 100 alle, jonka arvot ovat: koodi – 12003, joka on ensimmäinen vapaa koodi ikkunan nimikkeistöstä, nimi – Living room window, määrä – 3 ja yksikkö – kpl.

6. TOTEUTUSPROSESSI

Työn suunnittelun jälkeen on jäljellä itse työn toteuttaminen ja mahdollinen tuotteistaminen. Työkalut ja työtavat ovat tärkeä osa mitä tahansa ohjelmistoprojektia. Tässä luvussa käydään lyhyesti läpi kaikki työn toteutukseen liittyvät työkalut, työtavat ja prosessin vaiheet. Näiden lisäksi tässä luvussa käydään läpi lyhyesti työn tuloksena syntynyt ohjelma ja sen jatkokehitysmahdollisuudet.

6.1 Työkalut ja työtavat

Tämän työn lopputuloksena valmistunut ohjelma on toteutettu itsenäisesti tämän diplomityön kirjoittajan toimesta työn alussa tuotekehitystiimissä sovittujen vaatimusten mukaisesti. Työn toteutusta on havainnollistettu ja suunnitelmaa hyväksytetty väliajoin tuotekehitystiimissä. Tuotekehitystiimiltä on saatu myös neuvoja tarjouslaskentasovelluksen käyttöön ja ne otettu huomioon työtä tehdessä. Tiimillä oli myös viikoittaisia palavereja, joissa usein käytiin lävitse työn tila ja mahdolliset esteet, jos sellaisia oli.

Työkaluina kehityksessä käytin Azure Devops Spacen git-repositoriota versionhallintaan, Visual Studiota, Visual Studio Codea ja sen kanssa staattisessa testauksessa LINT-työkalua koodin tuottamiseen. Työn alussa päätimme olla käyttämättä ketterän kehityksen -työkaluja käyttäjätarinoiden, bugien ja muiden merkkaamiseen ja ajan tasalla pitoon, sillä työ toteutettiin hyvin itsenäisesti ja työn tilasta selvitykseen riittäisi demot aika-ajoin. Omaan dokumentaatioon, sekä käyttäjätarinoiden, tehtävien ja bugien kirjaamiseen käytin Microsoft Exceliä ja perinteistä ruutuvihkoa.

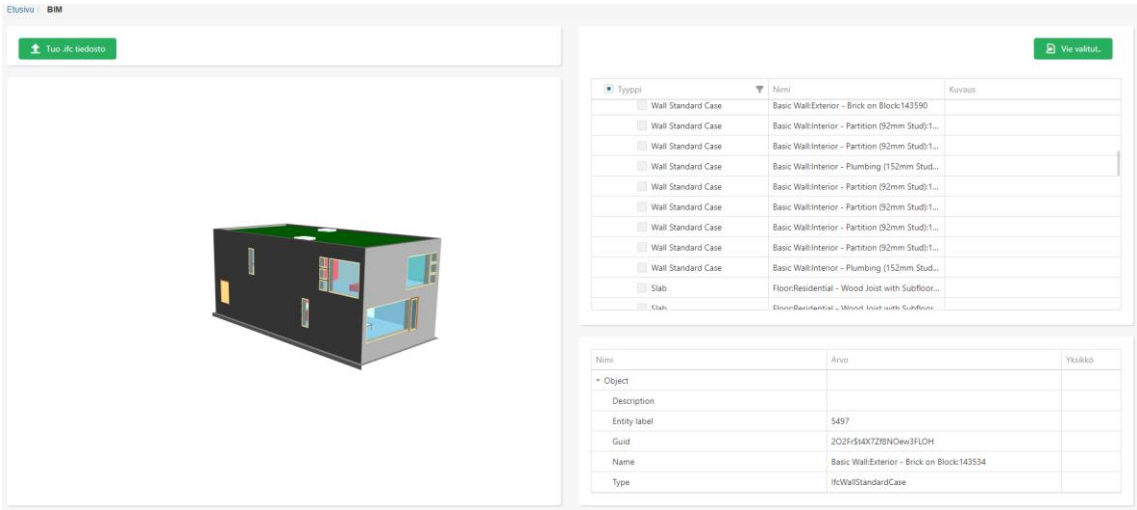
6.2 Prosessin kuvaus ja tulokset

Projektin alussa BIM-työkalu oli tarkoitus suunnitella ja toteuttaa vain verkkopohjaisena ratkaisuna upotettuna jo olemassa olevaan web-portaaliin. Ratkaisun teknologia valittiin alussa verkkopohjaiseksi sen ollessa enemmän modernia ja kätevämpää teknologiaa kuin työpöytäsovellukset. Tämä toteutusteknologia myös mahdollistaa nopean integroinnin sitten kun tarjouslaskentasovellus toteutetaan ja siirretään web-pohjaiseksi sovellukseksi.

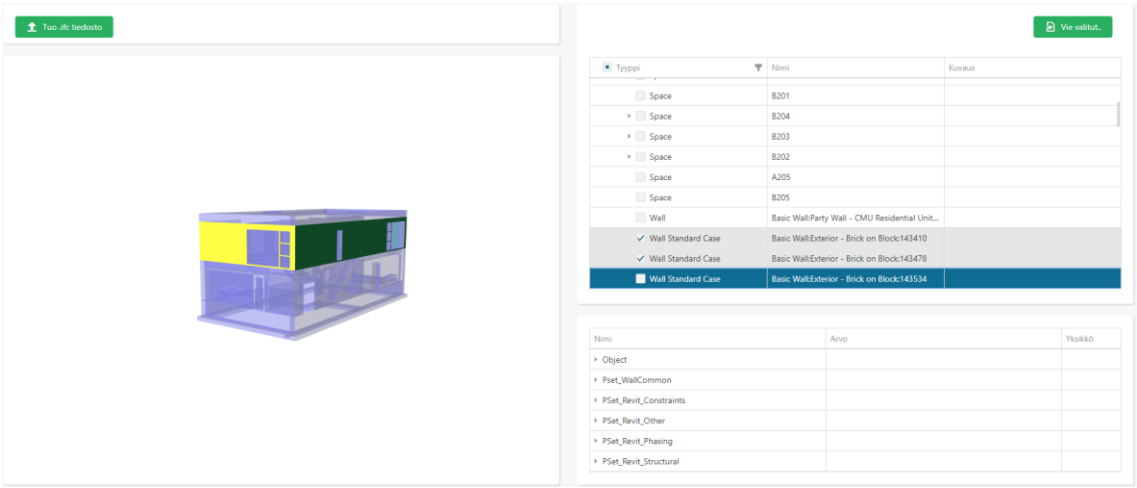
Suunnittelun ja toteutuksen edetessä tuli kuitenkin ilmi, että työkalu tulisi toimia ja kommunikoida upotettuna tarjouslaskentasovellukseen ilman erillistä verkkoselainta, sillä web-portaali ja tarjouslaskentasovellus ovat eri tuotteita. Tämä toi suunniteltuun ratkaisuun erinäisiä autentikaatio-, kommunikointi-, suorituskyky- ja käytettävyysongelmia, joten näimme järkevämmäksi ja nopeammaksi luoda kaksi erillistä sovellusta: verkkopohjaisen sovelluksen ilman integrointia toisiin sovelluksiin *proof-of-conceptiksi* ja työpöytäsovelluksen, joka liitetään nykyiseen tarjouslaskentasovellukseen.

Verkkopohjaisen sovelluksen asiakasohjelman komponenttiarkkitehtuuri suunniteltiin heti alusta lähtien samanlaiseksi mikä toimii Windowsin työpöytäsovelluksissakin, jotta työpöytäsovelluksen voisi toteuttaa samaa arkkitehtuuria käyttäen, mutta vain eri teknologialla. Toteuttaessa valittu xBIM-liitännäinen osoittautui hyväksi visualisoinnissa ja kolmiulotteisen mallin sai näkymään valmiiksi prosessoiduilla tiedostoilla hyvinkin nopeasti. Kuitenkin BIM-liitännäistä valittaessa ei otettu huomioon .NET core- ja .NET framework -ohjelmistokirjastojen eroja ja toteutusvaiheessa xBIM-liitännäinen ei uponnutkaan kirjastoksi palvelimelle. Tämä ongelma ratkaistiin luomalla yksityinen palvelin julkisen palvelimen rinnalle. Tästä työstä riippumatonta erillistä komponenttia varten oli jo valmiiksi toteutettu erillinen yksityinen palvelin tarvittavalla .NET framework -ohjelmointikirjastolla, joten se oli helppo kopioida myös tätä projektia varten.

Verkkopohjainen BIM-työkalu valmistui ensimmäisenä, josta näyttökuvat on esitetty kuvissa 30 ja 31. Kuvassa 30 on avattu talosta yksinkertainen malli ilman korostettuja objekteja ja kuvassa 31 on korostettuna seinä, jolloin talo muuttuu läpinäkyväksi selkeyden vuoksi.

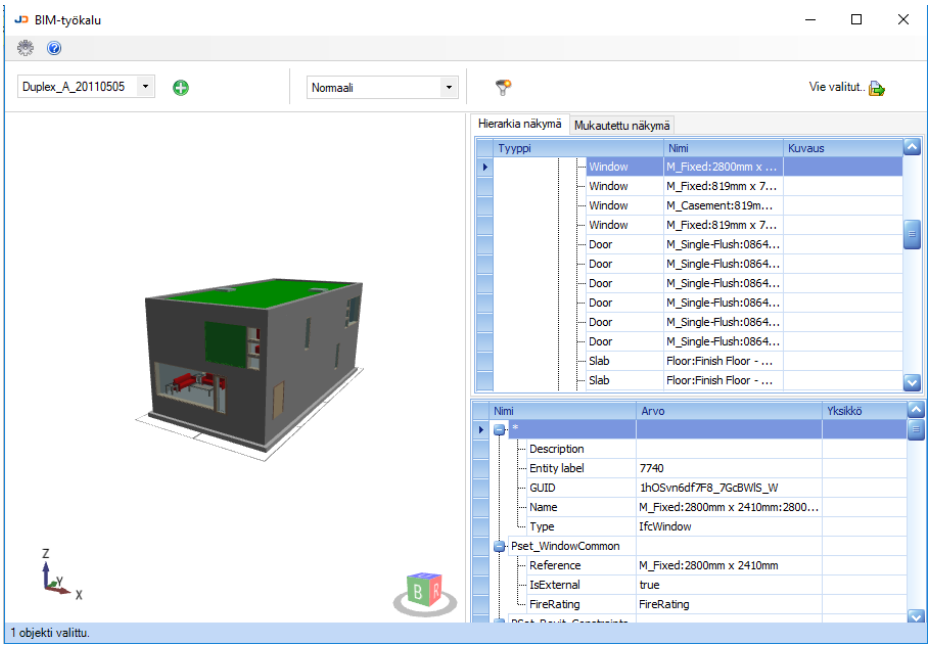


Kuva 30. Verkkopohjainen BIM-työkalu

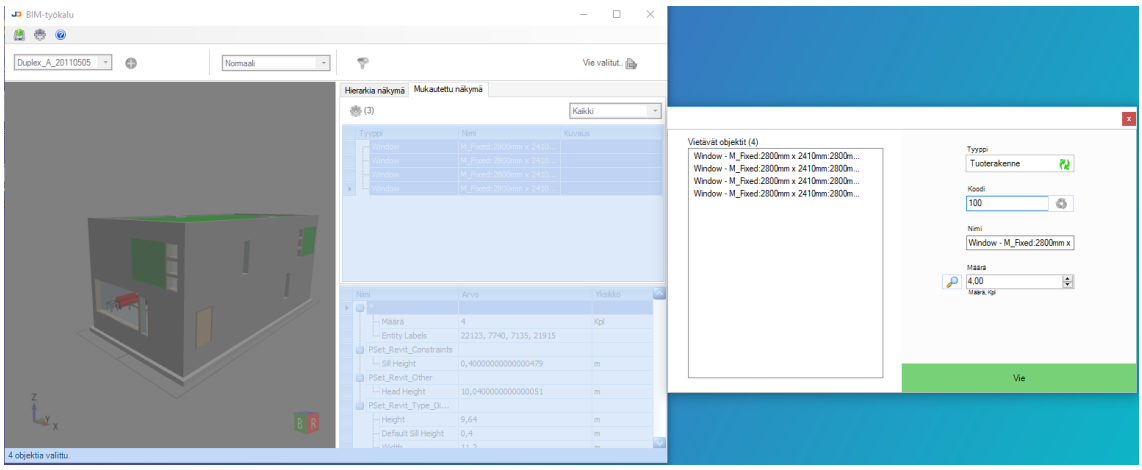


Kuva 31. Seinä korostettuna verkkopohjaisessa BIM-työkalussa

Verkkopohjaisen BIM-työkalun *proof-of-conceptin* valmistuttua Windowsin työpöydällä toimivan työkalun toteutus alkoi. Toteutettaessa työpöytäsovellusta ei ollut suurempia teknologisia esteitä. Ainoana suurempana esteenä oli se, että Jydacom Tarjouslaskenta -sovellus oli toteutettu käyttämällä 32-bittistä arkkitehtuuria ja BIM-mallien ollessa välillä hyvinkin suuria muistikooltaan, ohjelma kaatui usein sovelluksen muistin loppumiseen. Ratkaisuna tähän oli lukea BIM-mallit ohjelmaan pienemmissä paloissa, jolloin yhtäaikainen muistinkulutus ei ollut niin korkea. Näyttökuvat BIM-työpöytäsovelluksesta on esitetty kuvissa 32 ja 33. Kuvassa 32 on työkalun perusnäkymä ja kuvassa 33 on kuvattu vientikäsittelijän ikkuna, jolla objektien tietoja voidaan siirtää tarjouslaskentasovellukseen.



Kuva 32. Windowsin työpöydällä toimiva BIM-työkalu



Kuva 33. Objektien ominaisuuksien vienti tarjouslaskentaan

Kuten kuvista voidaan nähdä, verkkopohjainen BIM-työkalu ja työpöydällä toimiva BIM-työkalu näyttävät rakenteeltaan hyvin samalta.

6.3 Jatkokehitys

Koska sovellukset on toteutettu hyvin modulaarisesti, voidaan miettiä toisia sovelluksia, joissa olisi hyötyä BIM-työkalun keräämistä tiedoista. Esimerkiksi BIM työpöytä -sovellukseen voisi tehdä lisäosan, jolla voi viedä haluttuja tietoja ulos Microsoft Excel -taulukkolaskentasovellukseen. Myös web-sovellukseen voisi tehdä jonkinlaisen API-rajapinnan, jonka käyttäjä voi avata halutessaan jokaiselle omistamalleen mallille ja hakea rajapinnasta haluamiaan tietoja mallista omaan ohjelmaansa työpöytäsovellukseen toteutetun integraation tapaan. Tässä toki on hyvä ottaa huomioon autentikaatio ja muu turvallisuus.

Myös jonkinlainen reaaliaikainen yhteistyötila verkon ylitse voisi olla hyödyllinen, jos monta ihmistä haluaa käyttää sovellusta ja viedä objekteja toiseen ohjelmaan yhtäaikaisesti. Tällä hetkellä tämä on toteutettu tarjouslaskentasovelluksessa lukitsemalla työtila yhdelle henkilölle, joka myös vaikuttaa BIM-työkalun toimintaan samalla tavalla.

7. OHJELMAN ARVIOINTI

Ohjelman valmistuttua on tärkeä tarkastella ohjelman onnistumista alkuperäisten vaatimusten, sekä olemassa olevien ohjelmien ominaisuuksien valossa. On myös tärkeää arvioida olisiko ollut muita mahdollisia lähestymistapoja toteuttaa sovellus, sekä tiedostaa vaihtoehtoiset ohjelmat toteutetulle ohjelmalle. Tässä luvussa käydään lävitse edellä mainitut asiat sekä tarkastellaan ohjelman sekä itse teknologian potentiaalia tulevaisuudessa.

7.1 Vertailu alkuperäisiin vaatimuksiin

Työn alussa esitetyt tutkimuskysymykset on esitetty seuraavassa luettelossa.

1. ”Miten rakennuksen tietomallia hyödyntävän työkalun tulee toimia, jotta se olisi mahdollisimman käytettävä ja visuaalisesti selkeä”
2. ”Miten on mahdollista toteuttaa rakennuksen tietomallia hyödyntävä työkalu, josta pystyy välittämään halutun tiedon rakennuksen tietomallista tarjouslaskentasovellukseen”
3. ”Mitä muutoksia täytyy tehdä tarjouslaskentasovelluksessa olevaan arkkitehtuuriin, jotta se pystyy käsittelemään rakennuksen tietomallia hyödyntävältä työkalulta tullutta tietoa muuttamalla mahdollisimman vähän olemassa olevaa teknistä toteutusta”.

Ensimmäiseen tutkimuskysymykseen onnistuttiin vastaamaan luvuissa 3 ja 4. Luvussa 3 tarkasteltiin, kuinka BIM-työkalusta saadaan käyttökokemukseltaan optimaalinen muiden BIM-tarkasteluohjelmien ulkonäköä ja toimintoja vertailemalla. Luvussa 4 suunniteltiin näiden ja asiakasvaatimusten pohjalta BIM-työkalun BIM-mallin visualisointiin ja tarkasteluun tarkoitetut ominaisuudet. Nämä ominaisuudet toteutettiin sekä verkkopohjaiseen BIM-sovellukseen että työpöydällä toimivaan BIM-työkaluun. Kaikki aliluvussa 4.2 esitetyt toiminnalliset vaatimukset saavutettiin toteuttamalla sovellukset suunnitelmien mukaisesti.

Toiseen tutkimuskysymykseen onnistuttiin vastaamaan sekä verkkopohjaisen sovelluksen että työpöydällä toimivan sovelluksen osalta luvussa 5. Kuitenkin suunniteltu tiedonsiirto tarjouslaskentasovelluksen ja BIM-työkalun välillä toteutettiin vain työpöydällä toimivaan BIM-työkaluun. Työn alkuperäisenä teknologisena vaatimuksena oli selaimessa toimiva käyttöliittymä tarjouslaskentasovelluksen kanssa tietoa siirtävälle työkalulle, mutta tutkimuksen edetessä selvisi, että tämä ei ollut oikea lähestymistapa. Kaikki aliluvussa 5.2 esitetyt toiminnalliset vaatimukset saavutettiin työpöytäsovelluksen osalta. Työmäärältään lähestymistapa oli suurempi, mutta lopputulokseltaan parempi kuin alkuperäinen suunnitelma.

Kolmanteen tutkimuskysymykseen onnistuttiin vastaamaan aliluvussa 5.4. Tarjouslaskentasovellukseen lisättiin yhteistyömoduuli, joka mahdollistaa BIM-työkalun ja tarjouslaskentasovelluksen välisen kommunikoinnin. BIM-työkalun itsenäinen arkkitehtuuri ja yhteistyömoduuli mahdollisti sen, että tarjouslaskentatyökalun teknistä toteutusta joutui muuttamaan hyvin vähän sovelluksien välisen tiedonsiirron mahdollistamiseksi.

Toteutettua työtä voidaan pitää hyvin onnistuneena, sillä se vastaa kaikkiin alussa esitettyihin tutkimuskysymyksiin sekä täyttää alussa asetetut toiminnalliset vaatimukset. Ainut alkuperäinen vaatimus, jota työn toteutuksessa ei täytetty oli verkkoselaimessa toimivan sovelluksen ja tarjouslaskentasovelluksen välinen tiedonsiirto. Kyseinen vaatimus kuitenkin tutkimuksen ja työn edetessä kumottiin, joten se ei vaikuta työn onnistumiseen.

7.2 Vertailu olemassa oleviin ohjelmiin

Ei-toiminnallisina vaatimuksina oli käyttöliittymän samankaltaisuus olemassa oleviin ohjelmiin, käyttönopeus ja hyvä käyttökokemus. Käyttöliittymä onnistuttiin pitämään mahdollisimman samankaltaisena muihin BIMin tarkasteluun tarkoitettuihin sovelluksiin verrattuna, kuten esimerkiksi

Teklan BIMSightiin sekä BIM Visioniin. Käyttöliittymä onnistuttiin myös pitämään yksinkertaisena ja sitä kautta helposti käytettävänä. Käyttökokemus oli suurena prioriteettina BIM-työkalua tehdessä ja subjektiivisesta näkökulmasta työkalusta tuli hyvin intuitiivinen, nopea ja helppokäyttöinen. BIM-työkalun nopeutta on optimoitu indeksoimalla mahdollisimman paljon tietoa heti prosessointivaiheessa ohjelman taustalla, jotta sen lajittelutoimintoja olisi nopea ja sulava käyttää. Lajittelutoiminto pystyykin suodattamaan mallista melkein minkä tahansa suodatus- tai ryhmittelyehtojen perusteella alle kymmenessä millisekunnissa oikeat objektit, vaikka objekteja olisikin yli 15 000 ja ominaisuuksia yhteensä yli 900 000.

Taulukossa 2 on esitetty työkalun nopeus verrattuna muihin työkaluihin käyttäen samaa kuusi miljoonaa riviä sisältävää BIM-mallia kuin aliluvussa 3.1. Taulukossa on jaettu nopeus prosessoimattomaan nopeuteen sekä prosessoituun nopeuteen. Prosessoimaton nopeus tarkoittaa aikaa, joka kuluu ladatessa tiettyä BIM-mallia ensi kertaa ohjelmaan, jolloin malli prosessoidaan ja tallennetaan tietokantaan tai tiedostonhallintaan seuraavaa käyttökertaa varten. Prosessoitu nopeus tarkoittaa aikaa, joka kuluu mallin hakemiseen tiedostonhallinnasta ja näyttämiseen käyttäjälle. Testitietokoneena käytettiin samaa tietokonetta kaikilla ohjelmilla. Taulukossa kohdassa EVRY BIM-työkalu näytetään kaksi eri aikaa: normaali ja optimoitu. Optimoitu aika tarkoittaa sitä, että kolmiulotteisesta mallista jätetään osia lataamatta näytölle erittäin suuren mallin tapauksessa suorituskyvyn takaamiseksi. Normaali aika tarkoittaa mallin lataamista kokonaisuudessaan näytölle kerralla. Tässä taulukossa vertailtiin ainoastaan toteutetuista sovelluksista Windowsin työpöydällä toimivaa BIM-työkalua muihin sovelluksiin, sillä muut vertailtavat sovellukset olivat myös Windowsin työpöydällä toimivia.

	Prosessoimaton nopeus	Prosessoitu nopeus
Toteutettu BIM-työpöytäsovellus	Normaali: 4 min 45 s Optimoitu: 3 min 45 s	Normaali: 1 min 20 s Optimoitu: 20 s
Bim Vision	35 s	31 s
Tekla BIMSight	18 min	20 s
xBIM Xplorer	6 min	1 min 10 s

Taulukko 2. BIM-työkalun nopeuden arviointi

Kuten taulukosta voidaan nähdä, tuotettu BIM-työkalu ei yllä aivan parhaimmistoon prosessoimattomassa nopeudessa. Tämä on kuitenkin ymmärrettävää, sillä mallin prosessoinnin ytimenä on käytetty xBIM-ohjelmointikirjastoa, jota xBIM Xplorer myös käyttää. Verrattuna siihen prosessointiajasta on kuitenkin saatu leikattua optimoimalla käyttöliittymäkomponentteja, sekä säikeistämällä prosessia. Optimoidun mallin latausnopeus on parhaimmistoa, mutta siinä pitää ottaa tietenkin huomioon, että kolmiulotteista mallia ei ladata kokonaan.

Valmiiksi prosessoidun mallin näyttäminen oli samaa luokkaa kuin parhaimmistolla optimoidun mallin tapauksessa. Jos mallia ei optimoitu, prosessoidun mallin suorituskyky jäi kaikista huonoimmaksi vertailuun otetuista sovelluksista. Optimoidun ja normaalin mallin latauksen suorituskyvyn ero viestii selvästi siitä, että kolmiulotteisen mallin lataaminen on käytetyn xBIM-toolkit -ohjelmointikirjaston heikko kohta, sillä kolmiulotteisen mallin lataaminen käyttöliittymään on sen vastuulla. Yhteenvetona toteutetun työpöytäsovelluksen suorituskyky oli selvästi keskitasoa, joka oli yksi ei-toiminnallisista tavoitteista.

7.3 Vaihtoehtoiset toteutustavat

Toinen lähestymistapa toteuttaa työpöytäsovellus olisi ollut luvussa 3.2.1 kerrottu BIM Vision -API, joka olisi tukenut kaikkia vaadittavia ominaisuuksia. Huonona puolena tässä olisi kuitenkin ollut verkkokehittämisen tuen puuttuminen, eli sovellukset olisi joutunut jakamaan täysin kahteen erilliseen sovellukseen ilman yhteistä logiikkaa. Tämän lisäksi käyttäjän olisi tarvinnut avata ja asentaa kaksi erillistä ohjelmaa: Tarjouslaskentasovelluksen ja BIM Visionin käyttääkseen tätä toiminnallisuutta, sillä BIM Vision APIn käyttämiseen tarvitsee itse BIM Vision -ohjelman.

7.4 Kilpailu ja vaihtoehtoiset ohjelmat

BIM-työkaluja on markkinoilla paljon erilaisia ja tällekin työkalulle on tietenkin vaihtoehtoisia sovelluksia. Tällainen on esimerkiksi aiemmin mainittu Autodesk Revit. Revit tarjoaa rajapinnan yleisesti käytetyille kustannuslaskentaohjelmille ja pystyy välittämään tietoa niiden kanssa. Revitistä pystyy myös viemään BIM-mallissa olevat tiedot Microsoft Excelin käyttämään tiedostoformaattiin ja tekemään tarjouslaskennan Microsoft Excelissä. Mikä tahansa näistä vaihtoehdoista vaatii kuitenkin täydellisen poistumisen Jydacom Tarjouslaskenta -sovelluksesta ja luo tarpeen kouluttautua käyttämään uusia järjestelmiä. Myös aiemmin tarjouslaskentasovellusta käyttäneet joutuisivat siirtämään tiedot järjestelmästä manuaalisesti uuteen järjestelmään. Tässä työssä luotu BIM-työkalu pyrkii myös kilpailemaan näitä vaihtoehtoisia sovelluksia vastaan sen hyvällä käytettävyydellä, sillä se on suunniteltu pelkästään tähän tarkoitukseen. [2]

7.5 Ohjelman ja teknologian tulevaisuus

Tässä työssä tuotettu ohjelma tullaan liittämään osaksi tarjouslaskentasovellusta ja myymään sitä tuotteena. Tätä myötä ohjelman ylläpito ja jatkokehitys on välttämätöntä. Jos sovelluksesta halutaan saada suorituskyvyltään parempi, kolmiulotteisen mallin näyttävä komponentti olisi syytä tehdä uudelleen omalla toteutuksella tämän ohjelman tarkoituksia varten. BIM-teknologia on hyvin kypsä kaupallisille tuotteille, joten se on rakennusalalla todella nopeassa nousussa. BIM-teknologiaa käyttäviä sovelluksia tulee koko ajan enemmän ja sen käytön monimuotoisuus ja sen ympärillä olevat innovaatiot ovat jo nyt loistavia. Esimerkiksi Teklan BIMBuild -tapahtumassa esiteltiin BIM-tuotetta, jolla rakennusta rakennettaessa nähdään rakenneosien tarkat paikat lisätyn todellisuuden avulla [36]. Näen tämän työn kirjoitushetkellä BIM-teknologiasta tulevan välttämätön osa rakennuksien rakentamisprosessia ja käytön kasvavan entisestään tulevaisuudessa.

8. YHTEENVETO

BIM-työkalu suunniteltiin jo olemassa olevien BIM-tarkastelusovellusten pohjalta. Näiden perusteella suunniteltiin työssä suunnitellun ja toteutetun työkalun ulkonäkö, nopeus ja käytettävyys vaatimukset. BIM-teknologia on itsessään monimutkainen ja sisältää paljon erilaisia standardeja, joten ohjelmassa päädyttiin käyttämään xBIM-toolkit -ohjelmointikirjastoa BIM-mallien käytön mahdollistamiseksi työkalussa. xBIM-toolkit tarjosi työkalut IFC-tiedoston jäsentelylle, sekä tiedostosta jäsennellyn kolmiulotteisen mallin näyttämiselle. xBIM-ohjelmointikirjasto tuki web- ja työpöytäkehitystä, joten se sopi työhön erinomaisesti, eikä sen lisenssikään tuottanut ongelmia.

Työn alkuperäinen idea oli liittää BIM-työkalu EVRYn tuottamaan rakennusyritysten toimintaa ohjaavaan web-portaaliin ja liittää se jotenkin Windowsin työpöydällä toimivaan tarjouslaskentasovellukseen. Työn suunnittelun edettyä pidemmälle jouduttiin kuitenkin ohjelma jakamaan kahteen eri osaan: verkossa toimivaan BIM-työkaluun sekä Windowsin työpöydällä toimivaan työkaluun. Jako jouduttiin tekemään siksi, että tarjouslaskentasovellus ja web-portaali olivat eri tuotteita, eikä niitä voinut käyttää yhdessä käyttäjäystävällisesti. Jos tätä tuotteistusongelmaa ei olisi, ollut verkossa olevan BIM-työkalun ja tarjouslaskentasovelluksen kommunikointi olisi ollut mahdollista, joskin rajoitteista sekä hankalaa.

Verkossa toimiva työkalu tullaan liittämään osaksi verkossa olevaa EVRYn web-portaalia, joka on kehitetty käyttämällä .NET Core -ohjelmointikirjastoa sekä Angular-ohjelmointikirjastoa. Työpöydällä toimiva BIM-työkalu liitettiin osaksi tarjouslaskentasovellusta, joka oli toteutettu käyttämällä .NET Framework -ohjelmointikirjastoa sekä WinForms-ohjelmointikirjastoa. Yhteenveto käytetyistä teknologioista ja lopputuloksesta on esitetty taulukossa 3. Koska BIM-työkalu jouduttiin jakamaan kahdeksi eri sovellukseksi, jouduttiin toteuttamaan sille kaksi eri käyttöliittymää. Onneksi kuitenkin molempia teknologioita yhdisti sama logiikan toteutukseen käytetty ohjelmointikehys: .NET. Tämän vuoksi toteuttamalla BIM-työkalun logiikan hyvin, sitä pystyttiin käyttämään molempien työkalujen ytimenä ja logiikkana.

Lopputuotos	BIM-logiikka ja kaksi erillistä käyttöliittymää sille.
Käytetyt teknologiat	Työpöytäsovellus: .NET Framework, WinForms Verkkopohjainen sovellus: .NET Core, Angular
Tietokone arkkitehtuuri	Työpöytäsovellus: 32-bittinen Verkkopohjainen sovellus: 32- tai 64-bittinen
Käytetty kolmannen osapuolen kirjasto	xBIM-toolkit
Täyttää toiminnalliset vaatimukset	Kyllä
Täyttää ei-toiminnalliset vaatimukset	Kyllä
Käytettävyys	Hyvä
Suorituskyky	Keskitaso

Taulukko 3. Työn yhteenveto

Työn suunnittelun edetessä pidemmälle tarkentuivat ohjelman toiminnallisuus, arkkitehtuuri ja ulkonäkö. Molemmat BIM-työkalut suunniteltiin käyttämään mahdollisimman samaa kaavaa teknologioiden rajoitteet ja mahdollisuudet huomioon ottaen. Molemmat sovellukset saatiin toteutettua suunnitelmien mukaisesti ilman suurempia ongelmia.

Vaikka jouduttiin toteuttamaan kaksi erillistä sovellusta yhdelle logiikalle, molemmat sovellukset valmistuivat ajallaan suunnitelmien mukaan. Molemmat toteutetut sovellukset täyttävät kaikki niille asetetut toiminnalliset ja ei-toiminnalliset vaatimukset. Lopputuotoksena on kuitenkin kaksi erillistä ylläpidettävää käyttöliittymää yhden sijaan, mutta se on todennäköisesti parempi ja nopeampi vaihtoehto kuin yksi huonosti toimiva ja vaikeasti ylläpidettävä sovellus.

Toteutetun työpöytäsovelluksen käyttöliittymän käytettävyys ja suorituskykytestauksen myötä voidaan todeta, että toteutettu sovellus on käytettävyydessä vähintään muiden vastaavien sovellusten tasoinen sekä suorituskyvyltään sovellusten keskiluokkaa. Suorituskyvyn mittauksessa oli kuitenkin otettava huomioon, että BIM-tiedoston jäsentelyssä käytettiin kolmannen osapuolen ohjelmointikirjastoa, jonka nopeuteen ei voi vaikuttaa merkittävästi. Ohjelmasta tuli hyvä kokonaisuus, jonka kehitys ja tuotteistaminen jatkuu.

LÄHTEET

[1] G. Aranda-Mena, J. Crawford, A. Chevez, T. Froese, Building information modelling demystified: does it make business sense to adopt BIM? International Journal of Managing Projects in Business, Vol. 2, Iss. 3, 2009, pp. 419-434. http://www.emeraldinsight.com/doi/abs/10.1108/17538370910971063.

[2] BIM and Cost Estimating, in: Whitepaper, Autodesk.

[3] Autodesk Revit, https://www.autodesk.com/products/revit/overview.

[4] N.Č Babič, P. Podbreznik, D. Rebolj, Integrating resource production and construction using BIM, Automation in Construction, Vol. 19, Iss. 5, 2010, pp. 539-543. https://www.sciencedirect.com/science/article/pii/S0926580509001757.

[5] BIM GIANTS: A ranking of the nation's top BIM design and construction firms, 2016.

[6] BIMserver, in: http://bimserver.org/, 2019.

[7] buildingSMART IFC4x1 Final, http://www.buildingsmart-tech.org/ifc/IFC4x1/final/html/.

[8] BuildingSMART IFC, http://www.buildingsmart-tech.org/specifications/ifc-overview.

[9] buildingSMART IFC introduction, http://www.buildingsmart-tech.org/ifc/.

[10] ERDC:Duplex Apartment Model | ERDC - D-001, 2011.

[11] F. Chen, G. Mac, N. Gupta, Security features embedded in computer aided design (CAD) solid models for additive manufacturing, Materials & Design, Vol. 128, 2017, pp. 182-194. https://www.sciencedirect.com/science/article/pii/S0264127517304355.

[12] BIM Vision, https://bimvision.eu/en/download/, 2018.

[13] EVRY Jydacom Tarjouslaskenta EVRYltä, https://www.evry.com/fi/mita-teemme/services/ratkaisut/toiminnanohjaus-erp/jydacom/laskenta/.

[14] Free Software Foundation GNU Affero General Public License, https://www.gnu.org/licenses/agpl-3.0.html.

[15] Google Angular, https://angular.io.

[16] Google Angular architecture, https://angular.io/guide/architecture.

[17] Google Introduction to components, https://angular.io/guide/architecture-components.

[18] Google Introduction to services, https://angular.io/guide/architecture-services.

[19] D. Hallberg, V. Tarandi, On the use of open bim and 4d visualisation in a predictive life cycle management system for construction works, Journal of Information Technology in Construction (ITcon), Vol. 16, 2011, pp. 445. http://urn.kb.se/resolve?urn=urn:nbn:se:kth:diva-60984.

[20] Jungsik Choi, Yongha Lee, Inhan Kim, Development of Application for Generation of Automatic 2D Drawings based on openBIM, ISARC. Proceedings of the International Symposium on Automation and Robotics in Construction, IAARC Publications, Waterloo, pp. 1-6.

[21] H. Kim, K. Anderson, S. Lee, J. Hildreth, Generating construction schedules through automatic data extraction using open BIM (building information modeling) technology, Automation in Construction, Vol. 35, 2013, pp. 285-295. https://www.sciencedirect.com/science/article/pii/S0926580513000873.

[22] I. Kim, K. Kim, H. Kim, Z. Shen, A. Stumpf, J. Yu, BIM IFC information mapping to building energy analysis (BEA) model with manually extended material information, Automation in Construction, Vol. 68, 2013, pp. 183-193. https://www.sciencedirect.com/science/article/pii/S0926580516300656.

[23] A HIKE THROUGH POST-EJB J2EE WEB APPLICATION ARCHITECTURE, 2005.

[24] K.S. Malakhov, A.P. Kurgaev, V.Y. Velychko, Modern restful api dls and frameworks for restful web services api schema modeling, documenting, visualizing, PROBLEMS IN PROGRAMMING, Iss. 4, 2018, pp. 59-68. https://arxiv.org/abs/1811.04659.

[25] Microsoft Real-time ASP.NET with SignalR, https://dotnet.microsoft.com/apps/aspnet/real-time.

[26] Microsoft Registering an Application to a URI Scheme, https://docs.microsoft.com/en-us/previous-versions/windows/internet-explorer/ie-developer/platform-apis/aa767914(v=vs.85).

[27] Microsoft TypeScript, https://www.typescriptlang.org/.

[28] Microsoft What is .NET, https://dotnet.microsoft.com/learn/dotnet/what-is-dotnet.

[29] Microsoft Windows Forms, https://docs.microsoft.com/en-us/dotnet/framework/winforms/.

[30] C. Nance, TypeScript Essentials, Packt Publishing, Birmingham England], 2014.

[31] National Institute of Building Sciences National BIM standard, https://www.nationalbimstandard.org/faqs.

[32] BIMvie.ws, https://github.com/opensourceBIM/bimvie.ws, 2019.

[33] Pivotal RabbitMQ, https://www.rabbitmq.com/.

[34] How Instagram Feeds Work: Celery and RabbitMQ, 2013.

[35] R. Steinmetz, K. Wehrle, Peer-to-peer systems and applications, Springer, Berlin, 2005.

[36] Tekla BIMBuild event.

[37] Tekla BIMsight, https://www.tekla.com/tekla-bimsight/.

[38] xBIM Common Development and Distribution License, http://docs.xbim.net/license/license.html.

[39] xBIM Team xBIM Toolkit, http://docs.xbim.net/.

[40] XbimXplorer, http://docs.xbim.net/examples/xbimxplorer.html, 2018.

[41] xBIM, https://github.com/xBimTeam, 2007.

[42] xBimTeam xBimWebUI, https://github.com/xBimTeam/XbimWebUI.

[43] S.C. Yadav, S.K. Singh, An Introduction to Client/server Computing, New Age International, New Delhi, 2009.

[44] Yurii Boreisha, Oksana Myronovych, Publishing and Consuming RESTful Web API Services, Proceedings of the International Conference on Software Engineering Research and Practice (SERP), The Steering Committee of The World Congress in Computer Science, Computer Engineering and Applied Computing (WorldComp), Athens, pp. 104-110.

[45] S. Zhang, J. Teizer, J. Lee, C.M. Eastman, M. Venugopal, Building Information Modeling (BIM) and Safety: Automatic Safety Checking of Construction Models and Schedules, Automation in Construction, Vol. 29, 2013, pp. 183-195. https://www.sciencedirect.com/science/article/pii/S0926580512000799.